Universidad Complutense de Madrid

Trabajo de fin de grado del Grado en Ingeniería del Software. Facultad de informática

Gestión de la logística de exámenes en la Universidad

Autores:
Ignacio Domingo Martín
Julia Fernández Reyes
Ágata Sánchez Andreu

Director:
Simon Pickin

Curso 2018/2019

Resumen

El presente Trabajo de Fin de Grado ha sido realizado por tres alumnos especializados en el Grado de Ingeniería del Software y tiene como fin modernizar el proceso educativo mediante una mejora de la gestión de la logística de exámenes. Este proyecto se fundamenta en el estudio de necesidades y la implementación de una tecnología que facilite la organización y distribución de las aulas y los alumnos para la realización de los exámenes. La solución propuesta consiste en una aplicación web que ofrece a la administración la capacidad de gestionar aulas, fechas de exámenes y los respectivos modelos de examen de cada asignatura dentro de un solo sistema. Llegado el momento del examen, el proceso se llevaría a cabo mediante el envío a la aplicación de las lecturas de las tarjetas de estudiantes, de tal modo que la aplicación imprimiese y situase a los alumnos con el uso de un algoritmo de distribución aleatoria dentro del aula de tal modo que se reduciría la pérdida de tiempo en la organización de las aulas en el momento del examen y evita posibles situaciones irregulares de copia. Para realizar el cometido y hacerlo posible, se han tenido en cuenta las distintas restricciones y riesgos con los que contaría la aplicación dentro de su potencial entorno de instalación.

Palabras clave: Gestión, Logística, Exámenes, Universidad, Software, Aplicación Web, Arquitectura por capas, Node.js

Abstract

The main objective of the present end-of-degree project is to modernize the educational process by the study and development of a web application that pursues the facilitation of the examination logistics management. This project is based on the study of needs and the consequential implementation of a technology that facilitates the organization and distribution of classrooms and students for the realization of exams. The proposed solution consists of a web application that provides the administration with the possibility to manage classrooms, exam dates and the respective examination models of each subject within a single application. In the course of examinations, the process would be carried out by sending to the application the student card readings, in such a way that the application will print and place the students by means of a random distribution algorithm within the classroom, reducing the loss of time in the organization of the classrooms at the time of the examination and avoiding possible irregular situations of cheating. To carry out this task and make it possible, the different restrictions and risks with which the application would count within its potential installation environment have been taken into consideration.

Keywords: Logistics, Management, Examinations, University, Software, Web Application, Multitier Architecture, Node.js

Índice

Repositorio público

El código de la aplicación web que se ha desarrollado a raíz de esta memoria puede encontrarse en el siguiente repositorio público:

`https://github.com/nachodm/TFG`

1 Introducción

1.1 Perspectiva histórica

Desde el espectacular incremento en el número de universidades a lo largo de Europa durante la época medieval, los métodos de estudio y evaluación de los alumnos en las mismas han ido evolucionando a lo largo del tiempo. A comienzos de la Edad Media, el método de evaluación más común era los conocidos como *disputatio*[1], que consistían en la discusión oral de diversas materias destacando sus posibles aspectos a favor y en contra, con el objetivo de discernir lo que pudiera ser válido o verdadero y lo que no.

Esta metodología iría evolucionando con el paso del tiempo hasta convertirse en exámenes orales, y no sería hasta el siglo XVIII cuando las universidades europeas adoptarían los exámenes estandarizados escritos que en China se llevaban utilizando desde tiempos de la dinastía Sui (Siglo VI)[2]. Esta estandarización de los exámenes fue decisiva en el proceso de modernización de métodos de evaluación y enseñanza en el que estaban inmersos la mayoría de países europeos, ya que consistió en el primer método equitativo y racional para la evaluación de los estudiantes, así como servir como el primer método de acceso reglamentario a este tipo de estudios[1].

Sin embargo, durante estos últimos siglos la gestión logística de dichos exámenes en las universidades españolas apenas ha evolucionado; a pesar de estar ya inmersos en plena era digital, no se hace uso de las tecnologías a nuestro alcance, y la gestión de los exámenes sigue siendo en ocasiones sumamente ineficiente y sin estar ni mucho menos automatizada.

1.2 Situación actual

Se podría decir que la gestión actual de la logística de exámenes en la Universidad Complutense de Madrid y más concretamente en la Facultad de Informática cuenta con un amplio margen de mejora. La no automatización de dicha logística resulta en la mayoría de las ocasiones en un trabajo extra a realizar por el personal docente de la Universidad que se podría facilitar en muchos aspectos si se adaptase el proceso al entorno IT que actualmente nos rodea.

Actualmente, el sistema se guía por las siguientes etapas: la publicación del calendario de exámenes se lleva a cabo al comienzo del año académico, en el apartado *Exámenes por curso y grupo* de la sección de Estudiantes de la página web de la Facultad. Esta asignación de espacios se hace, en el caso de exámenes de teoría, sobre el número total de matriculados en la asignatura, y para los exámenes del laboratorio, teniendo en

[1] Hasta entonces, los alumnos ingresaban en las Universidades por medio de recomendaciones o impresionando a ciertos profesores con su conocimiento en diversas materias

cuenta estimaciones al alza sobre el número de alumnos que se espera se presenten al mismo; del mismo modo, la impresión del número de exámenes se realiza a la alza y con anterioridad al examen, transportándose posteriormente a las aulas con el consiguiente riesgo de extravío.

Este sistema provoca en ocasiones numerosos contratiempos como pueden ser:

- La pérdida de tiempo para el profesorado al tener que revisar, a la hora de reservar espacio en las aulas, cuántos de sus alumnos se podrían presentar a cada uno de sus exámenes.
- La posible reserva innecesaria de aulas que se pudieran quedar finalmente vacías, con los consiguientes gastos generados como pudieran ser el material de examen ya impreso, las horas de trabajo de preparación de laboratorios por parte de los técnicos, luz o calefacción.
- Pérdida de tiempo a la hora de colocar a los alumnos del modo que cada profesor considere oportuno dependiendo de cuántos alumnos se hayan presentado finalmente, sobre todo en los casos en los que se acaban juntando aulas.
- Posibles situaciones irregulares de copias, dada la capacidad de elección de cada alumno de sentarse en posiciones cercanas a sus compañeros más afines.

1.3 Motivación y alcance del proyecto

Ante el escenario descrito, el objetivo de nuestro grupo y de este TFG subyace en la posibilidad de usar la tecnología actual en la gestión de la logística de los exámenes, de modo que se pueda automatizar el proceso en la medida de lo posible, convirtiéndose de este modo en un sistema más seguro y eficiente. Consideramos que, mediante el uso de una aplicación web, el proceso de asignación de aulas, asientos y el de impresión de exámenes serían mucho más eficientes, a la par que asegurar en la medida de lo posible que no se podrá copiar en dichos exámenes mediante el uso de un algoritmo que asigne los puestos de forma aleatoria, dentro de las restricciones necesarias explicadas posteriormente. Esta aplicación deberá contar con acceso a la información relativa a las asignaturas y los alumnos matriculados en ellas, así como conocer el calendario de exámenes y la información del aforo de cada aula en la que se realizarán los exámenes.

Buscamos un sistema que pueda mejorar el funcionamiento de la Facultad en primer término, y que pueda ser expansible a otras facultades en un futuro próximo. Durante este proyecto se presentará dicha aplicación, profundizando en su funcionamiento, sus requisitos, sus riesgos, las tecnologías y lenguajes en los que se han desarrollado sus diferentes capas, su interfaz de usuario y los algoritmos que hemos podido usar en la misma.

1.4 Estructura del trabajo

La memoria de este Trabajo de Fin de Grado cuenta con un total de diez capítulos en castellano, incluyendo la introducción en la que nos encontramos, y dos adicionales en inglés, como son la Introducción y la Conclusión. El contenido resumido de estos capítulos es el siguiente:

- En el capítulo dos se encuentra la traducción al inglés de este primer capítulo de introducción.
- En el capítulo 3 se explica y describe el funcionamiento y el estado actual de la gestión de exámenes en entornos similares y el uso actual de tecnologías semejantes a la que se quiere implementar.
- En el capítulo 4 se encuentran la mayor parte de los diagramas que explican los requisitos y las características funcionales de la aplicación
- El capítulo 5 contiene una explicación detallada de las pruebas en entornos software y las que se han podido llevar a cabo dentro de nuestra aplicación.
- En el capítulo 6 se enumeran las tecnologías que han sido elegidas para realizar el desarrollo de la aplicación, así como los requisitos exteriores del entorno en el que se quisiera integrar. Se explica el funcionamiento de Node.js y las tecnologías relacionadas con dicho entorno.
- En el capítulo 7 se incluye una enumeración de los posibles riesgos con los que cuenta la aplicación, así como su estudio y la gestión que se pretende realizar de los mismos.
- El capítulo 8 describe detalladamente la implementación de la aplicación desarrollada, sus vistas, características y funcionamiento.
- En el capítulo 9 se describe la aportación de cada miembro del equipo a la realización de este Trabajo de Fin de Grado.
- En el capítulo 10 se describe el resultado logrado tanto en términos de investigación en esta memoria como en términos de implementación de la aplicación web, así como los conocimientos adquiridos por los miembros del grupo.
- En el capítulo 11 se presentan una síntesis final de este proyecto y un análisis de las líneas de desarrollo futuro que se podría realizar a partir del trabajo ya desarrollado.
- El capítulo 12 consiste en la traducción al inglés del capítulo inmediatamente anterior.

2 Introduction

2.1 Historical perspective

Since the remarkable growth in the number of universities throughout Europe during the medieval period, study and evaluation methods have evolved over time. At the beginning of the Middle Ages, the most common evaluation method was known as *disputatio* [1], which consisted in the oral discussion of a certain subject highlighting its possible positive and negative aspects, with the purpose of discerning which arguments could be valid and those that could not.

This methodology evolved over time to oral examinations, and it would not be until the eighteenth century when European universities adopted the standardized written examinations that the Chinese had been using since the Sui Dynasty (6th century) [2] This standardization of the exams was decisive in the process of modernization of evaluation and teaching methods in which most European countries were immersed, since it consisted in the first fair and rational method for the evaluation of the students, as well as serving as the first method of regulatory access to this type of studies[2].

However, during the last few centuries the logistics management of these exams in Spanish universities has hardly evolved; despite being already immersed in the digital era, we do not take advantage of the technologies we have available and the examinations management is still often extremely inefficient and not automated at all.

2.2 Current situation

It can be stated that the current examinations logistics management at the Complutense University of Madrid and more specifically in the Faculty of Computer Science has an ample room for improvement. The non-automation of this logistics results in several occasions in an extra job to be carried out by the teaching force of the university that could be facilitated in numerous aspects if the process was adapted to the IT infrastructure that surrounds us.

Currently, the organization scheme have the following stages: the examinations schedule is published at the beginning of the academic year in the *Exams by course and group* subsection of the Student section of the webpage of the Faculty. The allocation of classrooms is performed making upward calculations of the students that are likely to go to the exam over the total amount of students enrolled in each subject; in the same way, the estimates of the number of printed exams is done upwards and the printing is done immediately prior to the exam, being later transported by the teacher to the

[2] Until then, students entered universities through recommendations or impressing certain teachers with their knowledge in a subject

classrooms with the consequent risk of loss.

This process may cause numerous setbacks such as:

- The loss of time for teachers that have to make the estimates in the number of students for each of their subjects when reserving the classrooms.
- The possible unnecessary reservation of classrooms that could end up being empty, with the consequent expenses generated such as the already printed exam material, the hours of laboratories preparation work by technicians, electricity or heating.
- Loss of time when placing students during the exams in the way that each teacher considers appropriate depending on the number of students that finally take the exam, especially when it ends up gathering together students from different classrooms.
- Possible irregular situations of cheating, given the possibility of each student to sit close to their closest companions.

2.3 Project scope and motivation

Given this situation, our group pursues the possibility of using modern technology in the management of the logistics of exams, so that the process can be automated as much as possible, thus becoming a safer and more efficient system. We believe that, by using a web application, the process of assigning classrooms, seats and the printing of exams would be much more efficient, while ensuring where possible that you can not copy in these exams, by using a random students allocation algorithm, within the necessary restrictions explained later on this project. This application must have access to the stored information on the subjects and students enrolled in them, the exam calendar and the information of the capacity of each classroom in which examinations will be conducted.

Our aim is to create the necessary infrastructure to improve the operational process of the Faculty in the first place, and that can be expandable to other faculties in the near future. Throughout this document, the application will be presented, delving into the technologies and languages in which it has been carried out, its user interface, its internal functioning and the algorithms that we have used for it.

2.4 Project structure

This end-of-degree project has a total of ten chapters in Spanish, including the introduction, and two additional chapters in English, such as the Introduction we are in and

Conclusion. The summarized content of the incoming chapters is the following one:

- Chapter 3 explains and describes the current status of exam management in comparable environments and the current use of technologies similar to the one to be implemented.
- In Chapter 4 we can find most of the explanatory diagrams of the requirements and functional characteristics of the application.
- Chapter 5 contains a detailed explanation of the testing methods in software environments and those that have been carried out within our application.
- Chapter 6 lists the technologies that have been chosen to carry out the development of the application, as well as the external requirements of the environment in which it would be integrated. It explains the environment of Node.js and the technologies related to it.
- Chapter 7 includes an enumeration of the possible application risks, as well as its study and the management that it is intended to perform in order to handle them.
- Chapter 8 describes in detail the implementation of the developed application, its views, characteristics and features.
- Chapter 9 covers the contribution of each member of the team to the completion of this end-of-degree project
- In chapter 10 we describe the achieved results both in terms of this report's research and in terms of implementation of the web application, as well as the knowledge acquired by the members of the group during the process.
- Chapter 11 contains a final synthesis of this project and an analysis of the further development that could be made from the work already developed.
- Chapter 12 consists of the English translation of the preceding chapter.

3 Antecedentes y plan de trabajo

3.1 Introducción

Para poder realizar esta aplicación, previamente hemos tenido que informarnos sobre la experiencia en otras universidades. Tras haber estudiado los sistemas de éstas, hemos podido observar que el más desarrollado es el método de la Universidad Nacional de

Educación a Distancia (UNED), de la cual hemos obtenido que partes son aplicables a la UCM, y más en particular, a los exámenes de la Facultad de Informática. Claro está, el sistema de la UNED tiene aspectos que no son aplicables a una universidad presencial donde los exámenes están centralizados. Pero si la aplicación que se propone desarrollar está bien concebida y diseñada, si en algún momento la UCM decide implementar exámenes distribuidos (por ejemplo, ofreciendo sus grados en modo distancia), no debería ser difícil adaptarla.

Tras recoger información y haber iniciado una investigación sobre algunas plataformas oficiales de distintas universidades europeas, hemos podido observar que los métodos que se usan al realizar un examen son muy dispares a los establecidos en la Universidad Complutense de Madrid. A continuación, explicaremos algunas características que difieren en cuanto a la gestión de los exámenes.

3.2 Gestión de los exámenes en la UNED

En el año 2010 se empezaron a usar las primeras implementaciones de "La Valija Virtual"[3][4], consistentes en un proyecto de la UNED para realizar los exámanes presenciales con mayor rapidez y seguridad. Antes de la existencia de dicho proyecto, las pruebas se imprimían en Madrid y viajaban en papel en valijas cerradas con llave hasta cada centro asociado de la UNED. Cada profesor de la UNED tiene la obligación de pasar una semana como parte del tribunal vigilando exámenes en uno de los centros asociados, donde el destino se asigna por sorteo. Con el sistema anterior, el presidente de cada tribunal tenía que recoger en Madrid la llave de las valijas y viajar con ella hasta el centro asociado, con el riesgo de perderla.

Este innovador sistema facilitaba el registro del estudiante antes de realizar el examen. También determina el puesto que debe ocupar cada alumno en el aula, con la restricción de que un estudiante no se sitúe al lado de otro con la misma asignatura y nivel. De este modo se evitan las posibles copias entre los examinados. Una vez que el examen finaliza, el alumno lo entrega y el tribunal lo escanea, creando una copia de seguridad de todas las hojas, sistema que se usa para agilizar el proceso, permitiendo al profesor corregir los exámenes en pantalla al momento, y como garantía ante pérdidas[5] —no como antiguamente, que se tenía que esperar a que llegaran a la Sede Central—.

El principal objetivo de "La Valija Virtual" es poder reducir al máximo la espera de los alumnos para saber las calificaciones de sus pruebas, y que el estudiante disponga de una copia de su examen en la plataforma virtual. Otra ventaja es la minimización de costes y el ahorro de papel.

Este proceso controla digitalmente los exámenes. Se utilizan dos perfiles de usuario: aquellos pertenecientes al tribunal y personal del centro.

Con anterioridad a las sesiones de examen, la aplicación importa toda la información

referente al calendario de exámenes. Además, antes de iniciar la sesión, se realiza el descifrado de exámenes a través de la tarjeta universitaria del integrante del tribunal. A la entrada al examen, la aplicación permite asistir solo a los alumnos que tienen esa asignatura matriculada y que coincida con la hora y aula del examen. Un lector identifica el DNI o tarjeta del estudiante y una impresora imprime su examen. En la cabecera del examen, viene dado el puesto que debe ocupar cada estudiante y los materiales que puede usar para realizar la prueba. Para evitar colapsos, un sistema simula un semáforo, indicando cuando puede pasar el siguiente alumno. Durante la sesión, se realiza la identificación de la persona que ocupa cada puesto, además de controlar el tiempo que dispone cada estudiante con un sistema de avisos. También se consultan las relaciones de exámenes pendientes de entrega de cada asignatura y se obtiene la lista de los estudiantes que se han presentado. Por último, en la entrega de las pruebas, se necesita seleccionar la sesión del examen que está en curso. Mediante el escaneado de los exámenes, se registra la entrega y se envía una copia cifrada a la Sede Central.

3.3 Gestión de los exámenes en países de nuestro entorno

Durante el estudio de la información disponible sobre la logística de exámenes en distintas Universidades, nos encontramos con que un gran número de ellas que no habían publicado ningún tipo de información al respecto; dentro de las que presentaban documentación relativa al caso, las más completas para llevar a cabo la investigación resultaron ser las siguientes:

- Technical University of Denmark (Dinamarca)
- University of Oulu (Finlandia)
- University of Helsinki (Finlandia)
- Mälmo University (Suecia)
- Tallinn University (Estonia)
- University of Bergen (Noruega)

Todas las universidades estudiadas disponen de un plazo donde se exige al alumno realizar una preinscripción al examen que quiera realizar, “x” días antes del mismo. Si el alumno no se apunta a dicho examen dentro del horario establecido, éste no puede presentarse y pierde la oportunidad, por lo que se le sancionaría de una manera u otra. Por ejemplo, en la Universidad de Tallin [6], el alumno se marca como “ausente” en el examen. En el caso de la Universidad de Bergen [7], no correría convocatoria ni se sancionaría al alumno, por lo que, registrarse en el examen no afectaría. El registro para el examen también se puede cancelar en un plazo de tiempo concreto. En el caso

de las Universidades en las que nos hemos centrado en la investigación, la que tiene un plazo más amplio para registrarse/cancelar el registro en un examen es la Universidad de Tallin [6], donde se puede realizar la cancelación hasta un día antes del examen. Al contrario que en la Universidad de Helsinki [8], donde el plazo máximo es hasta 10 días antes del examen. Por último, si un alumno se registra para un examen y no se presenta, en la Universdidad de Tallin [6], éste suspende la asignatura, sin derecho a realizar la recuperación. En el resto de universidades citadas no se encuentra información para dicha situación.

Al igual que en estas universidades, nuestro objetivo es también incluir un método similar en nuestro proyecto. Esto contribuiría a distribuir más eficientemente las aulas de examen con los alumnos que se han registrado para realizarlo. En cualquier caso, estudiaremos realizar las pruebas con previo registro y sin él. Teniendo estos datos, podríamos juntar varios grupos con distintas asignaturas en una misma clase para realizar un examen. Además, facilitaría la asignación de los puestos de cada alumno en el aula dado que sabemos el número de personas que se presentarían al examen de cada asignatura, por lo que será necesario disponer de la tarjeta de la UCM e identificarse al entrar al aula del examen. Tras la identificación, el alumno podrá sentarse para realizar la prueba; si un alumno se registra para el examen y no se presenta, perdería una convocatoria para el examen, mientras que si el alumno no se registra para el examen, no puede presentarse, por lo que iría directamente a recuperación. Esto haría que los alumnos solo se registraran si se fueran a presentar. No obstante, siempre habría excepciones por falta debido a enfermedad, etc que puedan ser justificadas.

En estas universidades, los alumnos también tienen la obligación de identificarse. En la Universidad de Malmö [9] es necesario presentar la documentación para ingresar en el examen, mientras que en el resto de Universidades solo es necesario identificarse al final. También nos hemos informado de que algunas de estas universidades, realiza los exámenes en aulas con una gran capacidad de puestos. Por lo que realizan el examen muchos alumnos a la vez, como es la Universidad de Dinamarca [10]. Esto también podría ser una ventaja para nosotros, puesto que, si podemos juntar varios grupos en un aula, los exámenes se realizarían de una forma más rápida y el periodo de exámenes se podría acortar. También se podría prescindir de algunos profesores de guardia de examen en las aulas.

En estas universidades, los exámenes suelen realizarse en aulas, en el caso de la Universidad de Oulu [11], los asientos están asignados antes de iniciar el examen. Los laboratorios solo se usan en casos puntuales. Dado que en la Universidad Complutense de Madrid muchos exámenes son realizados en laboratorios, esto nos ayudaría mucho para distribuir a los alumnos, ya que, en muchas ocasiones se quedan laboratorios vacíos y hay que redistribuir a los alumnos de una misma clase para juntarlos a todos en el mismo laboratorio y este es el problema principal que evitaríamos.

3.4 Plan de trabajo

En cuanto al plan de trabajo realizado, hay que mencionar que la organización del trabajo será en todo momento un proceso iterativo, en el que nos permitirá arreglar fallos o modificar implementaciones que podrían estar mal. No obstante, se ha generado un Diagrama de Gantt [12] con las tareas que se van a ejercer y con el tiempo estimado que se va a tardar en hacer cada tarea. Hemos fragmentado el trabajo en dos partes: documentación e implementación. Tras esto, hemos indicado las tareas y subtareas correspondientes, así como la realización, las revisiones y las correcciones que podrían surgir.

A continuación, se muestra dicho diagrama, en el que se puede observar las distintas tareas, quién las ha realizado, las horas estimadas que se va a tardar en realizar cada tarea, el día de comienzo y fin estimado y por último, el porcentaje que se va realizando hasta que se finaliza. En la figura, además, se puede ver que hay tareas en verde, que estarían finalizadas, en rojas, que van en retraso en función de las horas estimadas; y en azul, que son las que aún faltan por realizar y están a tiempo.

Figura 1: Diagrama de Gantt

4 Análisis y especificación de requisitos

4.1 Enumeración

En primer lugar, cabe diferenciar los dos modos que podrá tener el proyecto: modelo dinámico y modelo estático, diferenciados únicamente por la posibilidad de una preinscripción al examen. Es decir, en el modelo dinámico no es necesario que los alumnos se preinscriban con antelación puesto que se les asignará el puesto correspondiente al llegar al establecimiento. Esto produciría problemas de optimización, que se podrán abordar con distintos algoritmos, intentando que la optimización de los puestos sea la mejor posible y cumpliendo una serie de restricciones. Uno de estos algoritmos podría ser el algoritmo del Simplex [13], cuyo objetivo principal sería maximizar los puestos en un aula. Otro tipo de optimización es la heurística, la cual se basa en minimizar o maximizar cierta función que contiene muchas variables a estudiar. No obstante, se pueden hacer estimaciones sobre el número de alumnos que van a presentarse mediante los profesores. Esto ocurre cuando alguna asignatura tiene requisitos imprescindibles para poder presentarse a un examen. Por ejemplo, la falta de alguna entrega práctica o la falta de asistencia a clase en un porcentaje mínimo.

En cuanto al modelo estático, será necesario que los alumnos se preinscriban. Para ello será necesario una plataforma para los alumnos que tenga los días de los exámenes y las asignaturas que tienen estos para poder presentarse. En el caso de que no haya restricciones en alguna asignatura para poder presentarse al examen, el número de alumnos a presentar será el total de la clase. No obstante, en ambos modelos habrá alguna estimación para poder especificar el número de puestos por clase que puede tener un examen. Todo esto implica que los requisitos se deban adaptar según la elección del modelo a realizar, siendo de esta forma necesarios o no algunos requisitos específicos.

Sin embargo, antes de distinguir los requisitos del proyecto hay que mencionar los distintos niveles de complejidad que se pueden desarrollar en la aplicación, ya que por tiempo no va a ser posible realizarlos todos. No obstante, indicamos dichos niveles con su posible implementación:

- **Grado de complejidad 1: Modelo estático.** En este nivel se implementará la funcionalidad más simple, de modo que solo exista la posibilidad de realizar en un aula un único examen de la asignatura, con un solo modelo, pero con la posibilidad de que efectúen el examen varios grupos.
- **Grado de complejidad 2: Modelo estático.** Este grado de complejidad abarca el primero, añadiendo la posibilidad de poder realizar varios modelos de examen de una misma asignatura. De modo que se podrá ubicar a los alumnos para que a su alrededor no tengan un compañero con el mismo examen.
- **Grado de complejidad 3: Modelo estático.** El tercer nivel contempla la

posibilidad de que en un aula se puedan ubicar dos o más grupos de asignaturas diferentes, pero con un solo modelo de examen en cada asignatura.

- **Grado de complejidad 4: Modelo estático.** Este grado comprende el tercer nivel en su totalidad con la adición de más modelos de examen en cada grupo y asignaturas distintas.
- **Grado de complejidad 5: Modelo dinámico.** En este nivel la implementación ya experimenta una dificultad mayor, pues al ser dinámico, el algoritmo va incrementando más problemas al ubicar a los alumnos en el aula. Así mismo, se podría tener un examen ubicado en una aula y con uno o varios grupos de la misma asignatura.
- **Grado de complejidad 6: Modelo dinámico.** En este grado de complejidad, la aplicación permite ubicar a los alumnos de distintos grupos en un mismo aula con más modelos de examen, pero de la misma asignatura.
- **Grado de complejidad 7: Modelo dinámico.** El penúltimo nivel consiste en tener en un mismo aula dos exámenes de distintas asignaturas, pero con un modelo de examen por asignatura.
- **Grado de complejidad 8: Modelo dinámico.** Este último grado de complejidad se corresponde con la idea de tener en un mismo aula alumnos que realizan distintos exámenes de distintas asignaturas, con distintos grupos y distintos modelos de examen por cada asignatura.

A continuación, veremos los distintos requisitos del proyecto según el modo que se quiera implementar. Estos se dividirán en: sistema, usuario, entorno y exámenes. No obstante, habrá requisitos comunes para ambos modelos. El proyecto necesitará tener un sistema que organice los exámenes en la facultad, así como las aulas/laboratorios donde se realizarán y los puestos de cada alumno según la asignatura y disponibilidad del aula. Por ello, los requisitos del entorno serán los siguientes:

1. Se podrá realizar varios exámenes de distintas asignaturas en una misma ubicación sin dejar huecos innecesarios, ya sea en el aula o en el laboratorio.
2. Juntar los alumnos de un aula con los de otra aula, una vez se viera que el número de presentados lo permita. Lo mismo ocurre con los laboratorios.
3. La ubicación de los alumnos en los puestos se hará con el objetivo de satisfacer ciertos criterios. Por ejemplo, en el caso de programar distintos exámenes (o distintos modelos del mismo examen) en el mismo aula, el algoritmo de asignación de puestos intentará asegurar que alumnos en puestos contiguos no tengan el mismo examen (o no tienen el mismo modelo del examen), sobre todo en el modelo dinámico, ya que en el estático se podrá predefinir previamente. Con esto se podrá garantizar el primer requisito.

Dado que los requisitos dependen también de la ubicación de los exámenes, como se describe en el punto 6.2, hay que distinguir ambos casos. En cualquier caso, los requisitos comunes son los siguientes:

1. Necesidad de un lector de tarjetas en aulas y laboratorios.
2. La aplicación tiene que tener acceso al sistema de la facultad para poder acceder a la información de los alumnos y las asignaturas de la facultad.
3. Disponer de impresoras rápidas suficientes en aulas y laboratorios, y disponer de escáneres rápidos en aulas.
4. Imprimir el examen de cada alumno, junto con las normas del examen.
5. El sistema deberá permitir enviar un aviso al móvil, o en su defecto mediante la aplicación, del profesor responsable del grupo de la asignatura si un alumno tiene alguna pregunta sobre el examen. Esto ocurre porque hay profesores de guardia en el examen que no imparten la parte teórica o que, incluso, no imparten la asignatura de este.
6. La interfaz del sistema deberá tener una representación espacial del aula o laboratorio con una imagen del alumno, su nombre completo y la asignatura del examen que realizará.
7. La interfaz deberá tener un apartado de "Notas" para que un profesor de guardia pueda indicar si ha habido alguna incidencia con algún alumno o para describir las dudas que pueda tener durante el examen, siempre y cuando esté permitido hacer preguntas. De esta manera, quedará reflejado para que el profesor responsable que corrija dicho examen pueda tomar ciertas medidas o no.

En cuanto a los requisitos específicos del sistema en los laboratorios, tenemos que:

1. El sistema deberá poder interactuar con las aplicaciones correspondientes de entrega de exámenes que posee la Facultad de Informática para permitir proporcionar ficheros a los alumnos al principio del examen y para permitir la entrega de ficheros al finalizar el examen.
2. El examen tiene un tiempo programado y se cerrará después de este, pero el profesor de guardia tiene la posibilidad de posponer la entrega. No obstante, tendrá que autenticarse al salir del laboratorio.

Por otro lado, los requisitos específicos del sistema en las aulas serán los siguientes:

1. La interfaz del sistema permitirá al profesor de guardia indicar si el alumno ha decidido entregar o no el examen, aunque haya realizado los ejercicios. Esto es debido a que un alumno puede decidir si lo presenta para que se lo corrija el profesor responsable o, si por el contrario, no quiere gastar una convocatoria. Todo esto siempre y cuando esté permitido "no entregar el examen".

2. Al salir de un examen, el alumno tendrá que autenticarse para indicar que se ha presentado y a qué hora ha salido del examen. Esto remplazaría las firmas que piden algunos profesores en los exámenes, sobre todo en los que se realizan en los laboratorios.

Los requisitos del usuario serán imprescindibles para que el sistema funcione correctamente. Por ello los requisitos son:

1. Los usuarios tendrán que estar dados de alta en la base de datos del sistema, mediante su carné universitario. En el caso de que un alumno no tenga la tarjeta, el profesor de la asignatura decidirá si se puede presentar enseñando su DNI. Si se da el consentimiento, el profesor añadirá a través de la vista a dicho alumno e imprimirá el examen desde un botón de la interfaz.

2. Necesidad de tener un carné universitario. En el caso de los alumnos Erasmus, se les proporcionará ese carné al igual que el resto.

3. El día del examen el alumno deberá autenticarse, por medio del carné, en el aula correspondiente. De esta manera se le podrá asignar su puesto.

4. En el caso del modelo estático, el alumno tendrá que preinscribirse al examen antes del día en que se realice.

5. El alumno tendrá que pasar la tarjeta al entrar en el examen y al salir para que quede registrada la hora de comienzo y fin.

6. Si el alumno tiene alguna duda y está permitido hacer preguntas durante el examen, este tendrá que solicitar ayuda al profesor de guardia.

7. Si el alumno decide no entregar el examen y está permitido hacerlo, tendrá que comunicárselo al profesor de guardia.

8. El profesor de guardia deberá comprobar el aula, así como los puestos ocupados y la identificación correcta de cada alumno.

9. Si durante el examen surge alguna duda de algún alumno y no puede resolverla el profesor de guardia, este se encargará de enviarle un mensaje al profesor responsable del grupo para que pueda acudir y resolver las dudas. Teniendo en cuenta que esta opción está permitida.

10. Si durante el examen ocurre alguna incidencia deberá escribir en la aplicación, concretamente en el apartado de "Notas", dicha incidencia y resolverla si fuese necesario.

11. Si la entrega se realiza en papel, al finalizar un alumno su examen, el profesor de guardia tendrá que recoger dicho examen y anotar si ha entregado o no. Esta última parte solo será necesaria si es posible "no entregar".

12. El profesor responsable tendrá que revisar que todos los exámenes han empezado, acudir a resolver las dudas que surgiese y, al finalizar, recoger los exámenes del aula o laboratorio.

13. En el modelo dinámico será necesario que los alumnos se identifiquen dos veces en el lector de tarjetas, una antes de comenzar el examen y otra una vez que estén los puestos asignados. Ya que la asignación puede cambiar si se decide, por ejemplo, juntar varios aulas.

Por último, los profesores responsables de cada asignatura serán los encargados de cumplir con los requisitos de sus exámenes. Al igual que los requisitos del usuario, estos son imprescindibles, ya que, si se hacen mal el sistema no podrá funcionar correctamente. Los requisitos son los siguientes:

1. El profesor deberá indicar el número de modelos que tendrá el examen.

2. El examen tendrá que ser un formato editable para que al imprimir los exámenes se añada el nombre, los apellidos y el puesto del alumno correspondiente. Una buena elección sería un PDF editable o un DOC/DOCX.

3. En el caso de la existencia de una preinscripción, habrá que hacer un recuento de los alumnos que van a presentarse al examen. En el caso de sin preinscripción, el profesor responsable de un grupo tendrá que facilitar la información del número de alumnos que podrán presentarse, así como dar una estimación a la alza de los presentados a dicho examen.

4. Para el correcto funcionamiento de la base de datos, es necesario que al crear un evento se le llame con el formato NombreDeLaAsignatura-MesAño, y para subir un examen a la plataforma, es necesario que siga el mismo formato pero añadiendo el modelo del examen, NombreDeLaAsignatura-MesAño-ModeloX. Por ejemplo, evento: "Base de datos-Junio2019" y examen que se realiza en esa fecha: "Base de datos-Junio2019-Modelo1".

5. En el modelo dinámico, será necesario que los profesores responsables de un grupo informen del número de alumnos que podrán presentarse, realizando una estimación a la alza.

4.2 Procesos

En este apartado se van a poder visualizar mejor los procesos que tiene que realizar cada actor. Hay dos procesos que se van a desarrollar: Proceso de preparación de examen, realizado antes de que comience el día del examen, y el Proceso de examen que se realiza durante el día del examen. Seguidamente, se muestra el diagrama completo, con todas las relaciones y subprocesos que son necesarios. Luego, aparecerán los distintos subprocesos desplegados para que se visualicen mejor y se pueda entender el proceso completamente. Cabe indicar que los procesos se podrían haber hecho con un *pool* y varios *swimlanes* o varios *pool*. Sin embargo, tras un estudio de ambos casos, nos pareció más conveniente optar por la primera opción, la cuál se compone de un *pool*, indicando el proceso que se va a realizar y dentro de ellos tres *swimlanes* con los actores que intervienen en dicho proceso.[3]

El proceso comenzaría con un alumno pasando la tarjeta por el lector. En el caso de no tener tarjeta, se tiene que identificar mediante otro documento de identidad, ya sea el DNI o pasaporte, y el profesor de guardia lo verifica y lo ingresa en la aplicación. Una vez hecho esto, si el alumno tiene examen ese día y no ha llegado tarde (ha pasado más tiempo del permitido), se le asigna el puesto, se imprime el examen y el alumno se sienta. El alumno comienza el examen (a la par, el profesor responsable comprueba si ha empezado el examen en las aulas) y el profesor de guardia lo vigila. Durante el proceso de realización del examen, al alumno le pueden surgir dudas, cuando esto pasa, el alumno avisa al profesor de guardia de su aula y éste, si no puede resolver el problema y si no es el profesor responsable, envía un mensaje al profesor responsable avisándole sobre que un alumno tiene alguna duda o si ha surgido algún imprevisto. Si el profesor responsable decide cambiar la hora de finalización del examen, envía un mensaje a los profesores de guardia y estos informan a los alumnos de su aula. Un alumno puede salir del examen si ha pasado X tiempo (donde X es mayor que el tiempo máximo de retraso de entrada permitido) desde la hora de comienzo del examen. Una vez un alumno entrega el examen, el profesor de guardia apunta si ha entregado y a la salida, el alumno, vuelve a pasar su tarjeta por el lector para que quede constancia de que ha finalizado su examen. Una vez todos los alumnos finalizan el examen, el profesor responsable los recoge (de una manera u otra dependiendo si la entrega es electrónica o en papel) y el proceso finaliza. Todo este proceso se puede ver formalizado en los siguiente diagramas.

Al final de dichos diagramas aparecen varias tablas explicando el significado de las señales, ya que se dividen en varios tipos: S (Evento de Señal), M (Evento de Mensaje), C (Evento condicional), T (Evento de Tiempo). Estas señales son enviadas y recibidas por distintos procesos o actores.

[3]Esta representación se ha diseñado por medio de el lenguaje estándar BPMN [14]

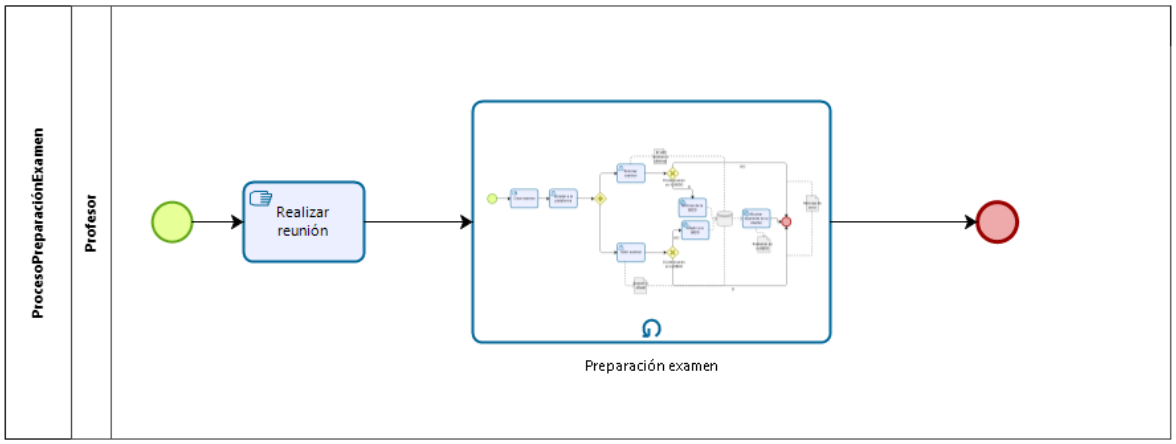

Figura 2: Proceso Preparación Examen

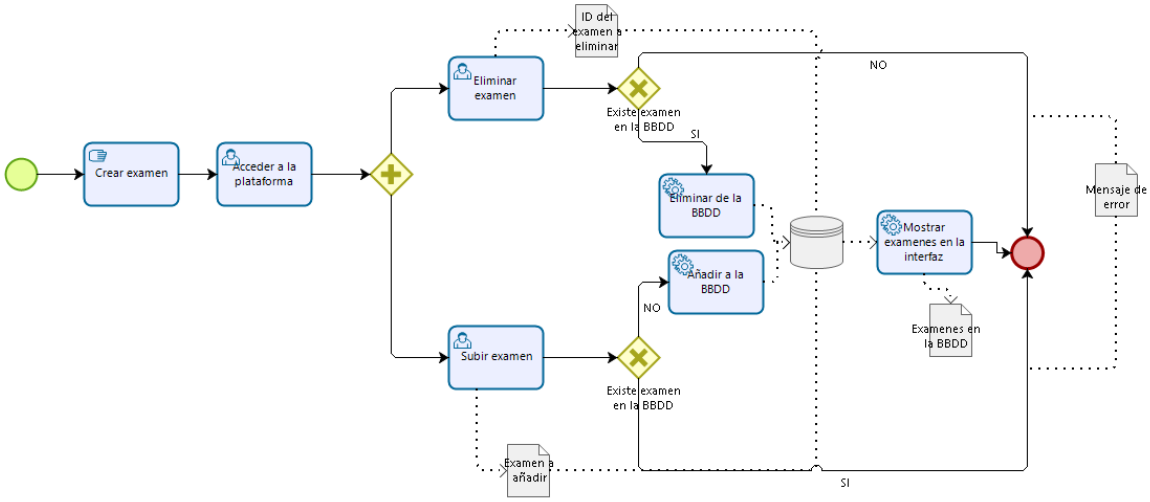

Figura 3: Subproceso Preparación Examen

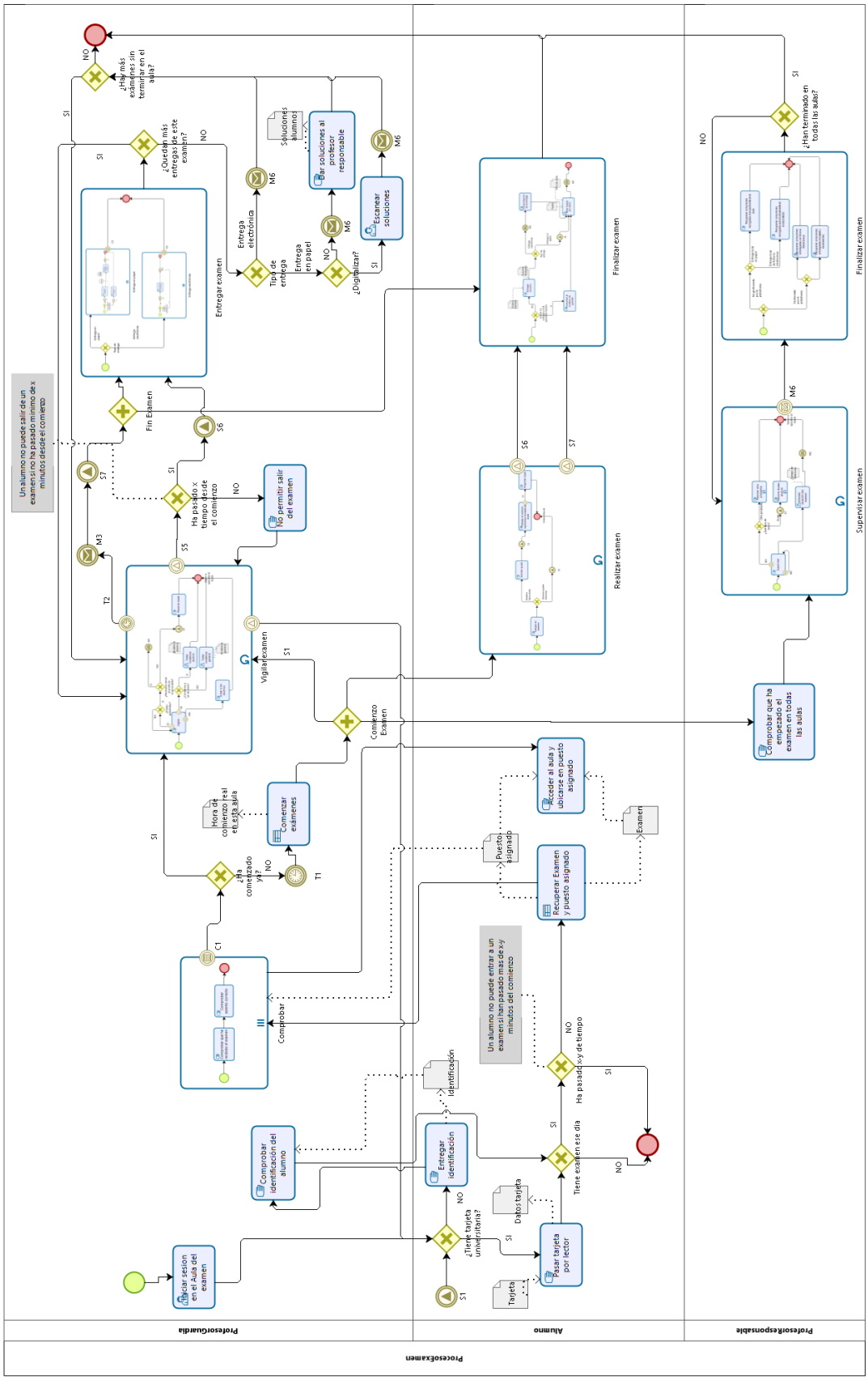

Figura 4: Proceso Examen

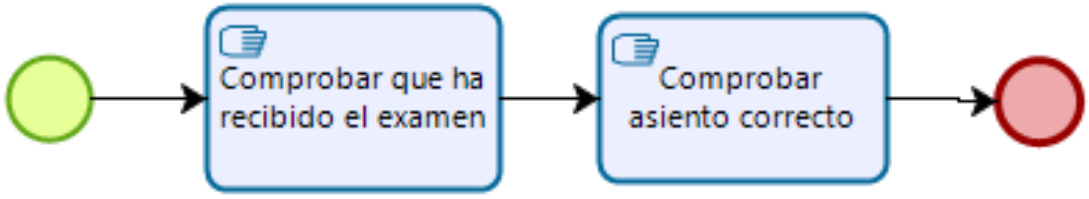

Figura 5: Subproceso Comprobar del profesor de guardia

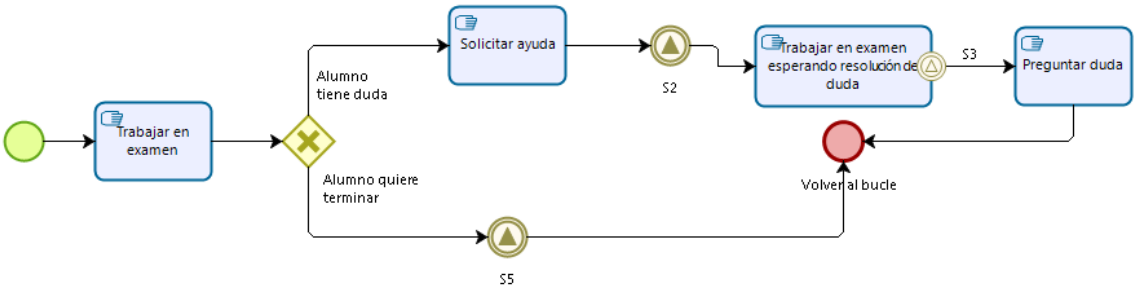

Figura 6: Subproceso Realizar Examen del alumno

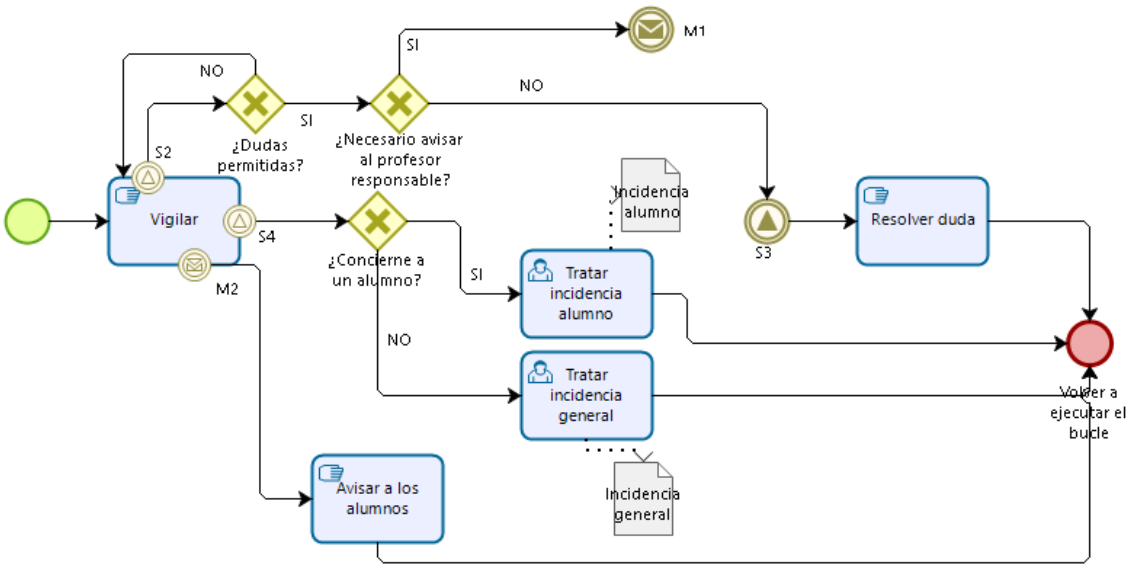

Figura 7: Subproceso Vigilar Examen del profesor de guardia

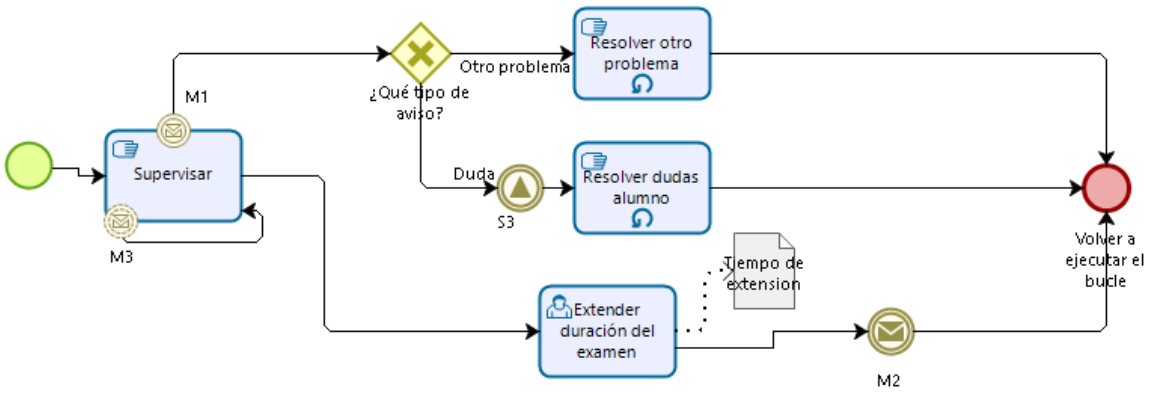

Figura 8: Subproceso Supervisar Examen del profesor responsable

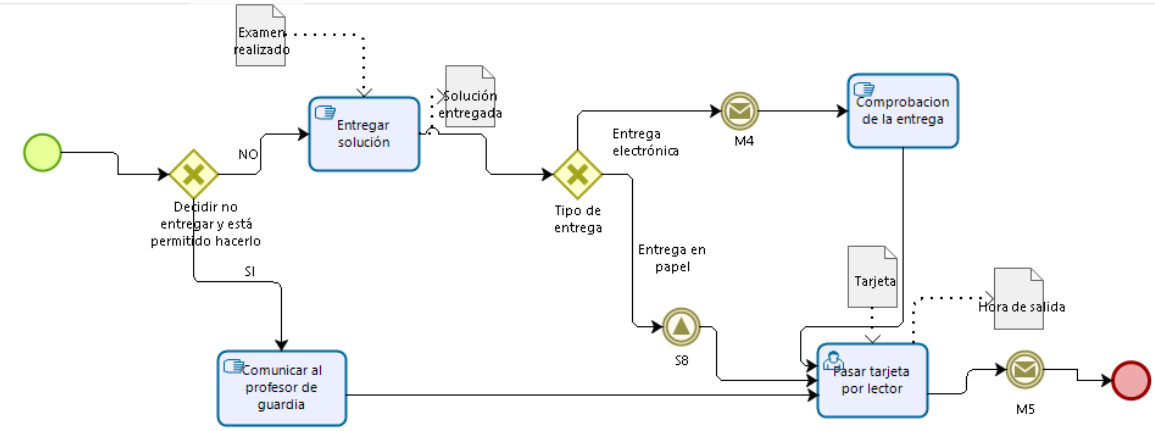

Figura 9: Subproceso Finalizar Examen del alumno

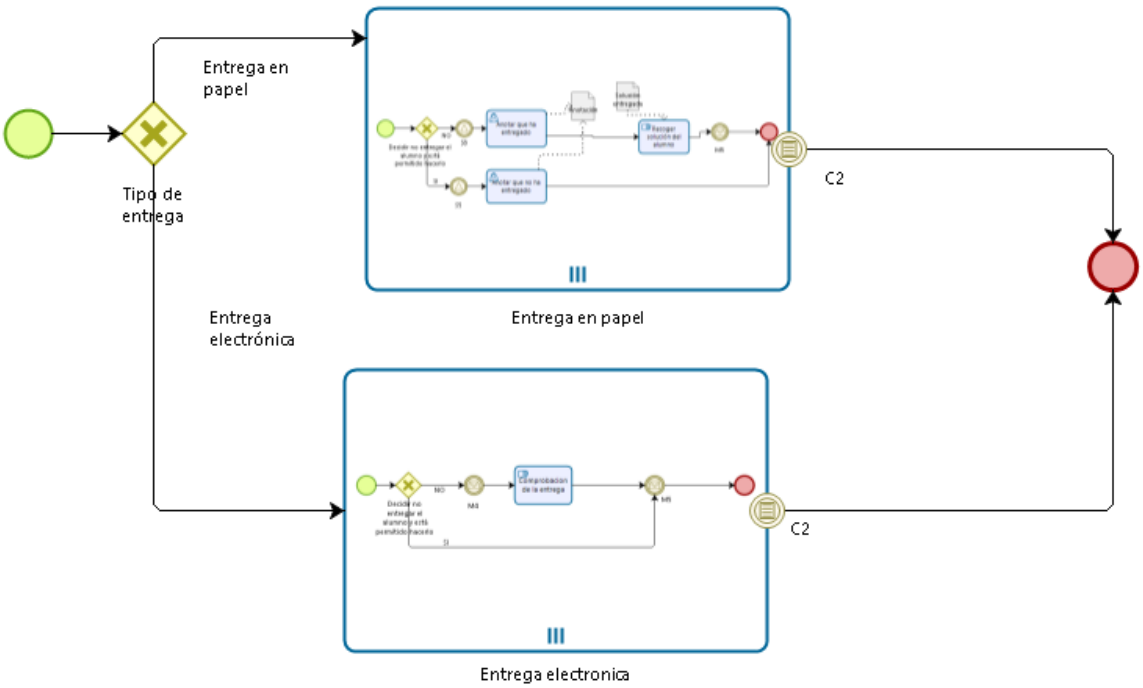

Figura 10: Subproceso Entregar Examen del profesor de guardia

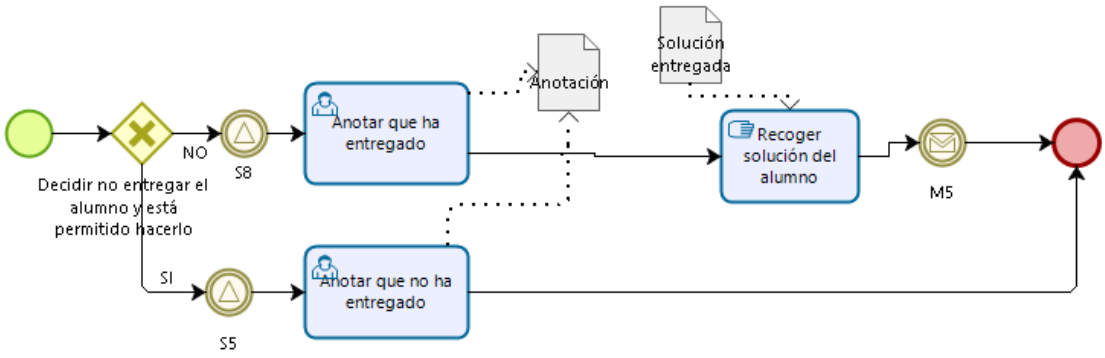

Figura 11: Sub-subproceso Entrega en Papel del profesor de guardia

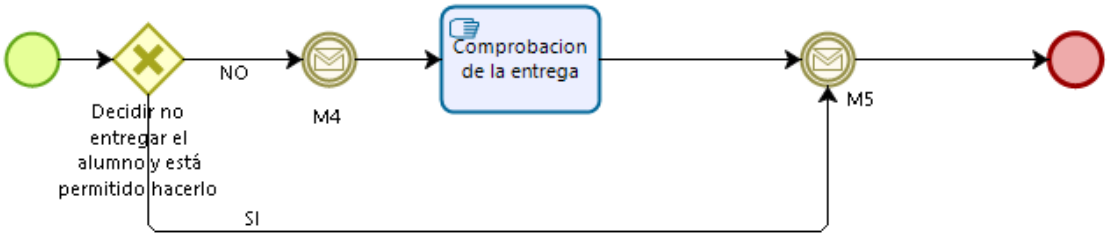

Figura 12: Sub-subproceso Entrega Electrónica del profesor de guardia

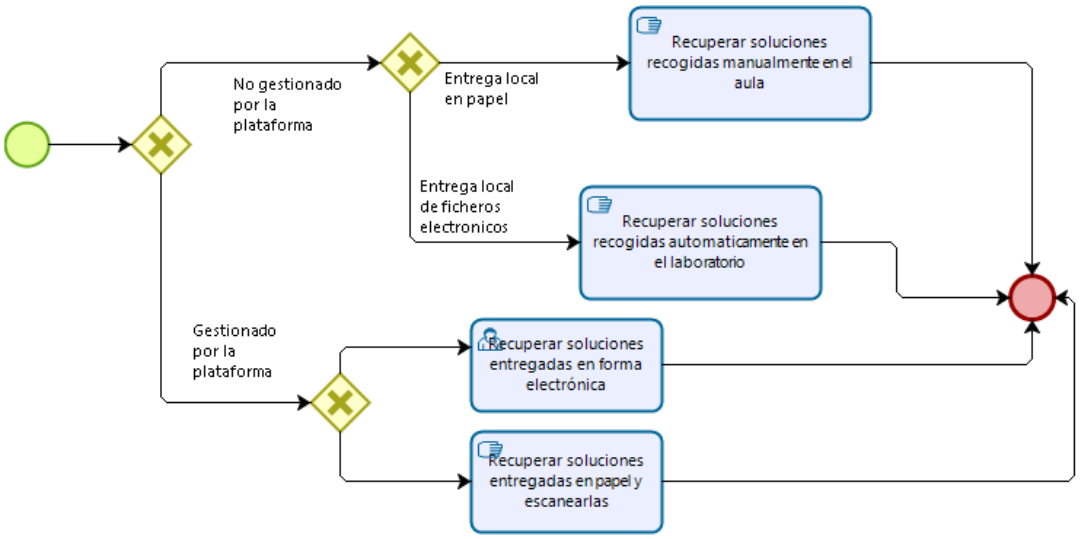

Figura 13: Subproceso Finalizar Examen del profesor responsable

EVENTOS DE SEÑAL:

N.º	Nombre	Lanzado por	Recogido por
S1	Llegada alumno	Alumno	El profesor de guardia en el aula (si el examen ha empezado) o nadie (si no)
S2	Duda alumno	Alumno	profesor de guardia del aula
S3	Resolución duda	Profesor de guardia o el profesor responsable	Alumno que tiene la duda
S4	Incidencia	Entorno del sistema	Profesor de guardia
S5	Petición salida	Alumno	Profesor de guardia del aula
S6	Autorización salida	Profesor de guardia	Alumno que quiere salir
S7	Fin examen	Profesor de guardia	alumnos del aula que están realizando el examen que termina
S8	Entrega solución en papel	Alumno	Profesor de guardia del aula

Figura 14: Leyenda de eventos de señal

EVENTOS DE MENSAJE:

N.º	Nombre	Enviado por	Recibido por
M1	Duda alumno	Profesor de guardia	Profesor responsable de este examen
M2	Extensión examen	Profesor responsable de un examen	Uno o varios profesores de guardia de las aulas de este examen
M3	Fin de examen	Profesor de guardia	Profesor responsable de este examen
M4	Entrega solución electrónica	Alumno	Profesor de guardia del aula
M5	Tarjeta pasada	Alumno	Profesor de guardia del aula
M6	Soluciones recogidas	Profesor de guardia	Profesor responsable de este examen

Figura 15: Leyenda de eventos de mensaje

EVENTOS DE CONDICIÓN:

N.º	Nombre	Descripción
C1	No hay más alumnos en la entrada	Cierto cuando no hay más alumnos en cola para entrar.
C2	No hay más alumnos en la salida	Cierto cuando no hay más alumnos en cola para salir.

Figura 16: Leyenda de eventos de condición

EVENTOS DE TIEMPO:

N.º	Nombre	Descripción
T1	Hora oficial de comienzo	La hora oficial de comienzo de todos los exámenes que están teniendo lugar en el aula donde está el profesor de guardia.
T2	Hora fin de un examen	La hora de terminación de un examen (hora de comienzo real de todos los exámenes del aula + duración oficial del examen + extensión, si el profesor responsable ha dado una) que está teniendo lugar en el aula donde está el profesor de guardia.

Figura 17: Leyenda de eventos de tiempo

4.3 Modelo de dominio

El modelo de dominio es un artefacto de la fase de análisis que proporciona una visión abstracta de las entidades implicadas/involucradas en la aplicación y las relaciones entre ellas. Suele estar muy relacionado con el modelo de datos, un artefacto de la fase de diseño de alto nivel que incluye algunas detalles de diseño, que a su vez está muy relacionado con el esquema de la base de datos, un artefacto del diseño de bajo nivel (que incluye datos detallados como cuál es la clave primaria, etc.)

En el Modelo de dominio[15] que hemos creado, se puede observar que hay varias entidades, no obstante, las que más relevancia tienen son las siguientes: Por un lado está el *Actor Universitario*, que se separa a su vez entre profesor y alumno. Por otro lado está el *Registro docente*, que se relaciona con todo lo que ello conlleva. Y por último, está la entidad de *Examen* con sus respectivas relaciones. Todas ellas son necesarias vincularlas con más clases que hacen que el sistema tenga consistencia. En el caso de la entidad *Aula* será necesaria para poder realizar las distintas interfaces para la aplicación y para reflejar la relación entre "ReservaAula" dirigida por un profesor y "Asiento" que ocupa un alumno. Además, en la siguiente figura también se puede observar todas las multiplicidades que tienen las distintas entidades, así se puede tener un concepto más amplio sobre la aplicación. Cabe destacar que cuando la relación no tiene un número, la multiplicidad es 1.

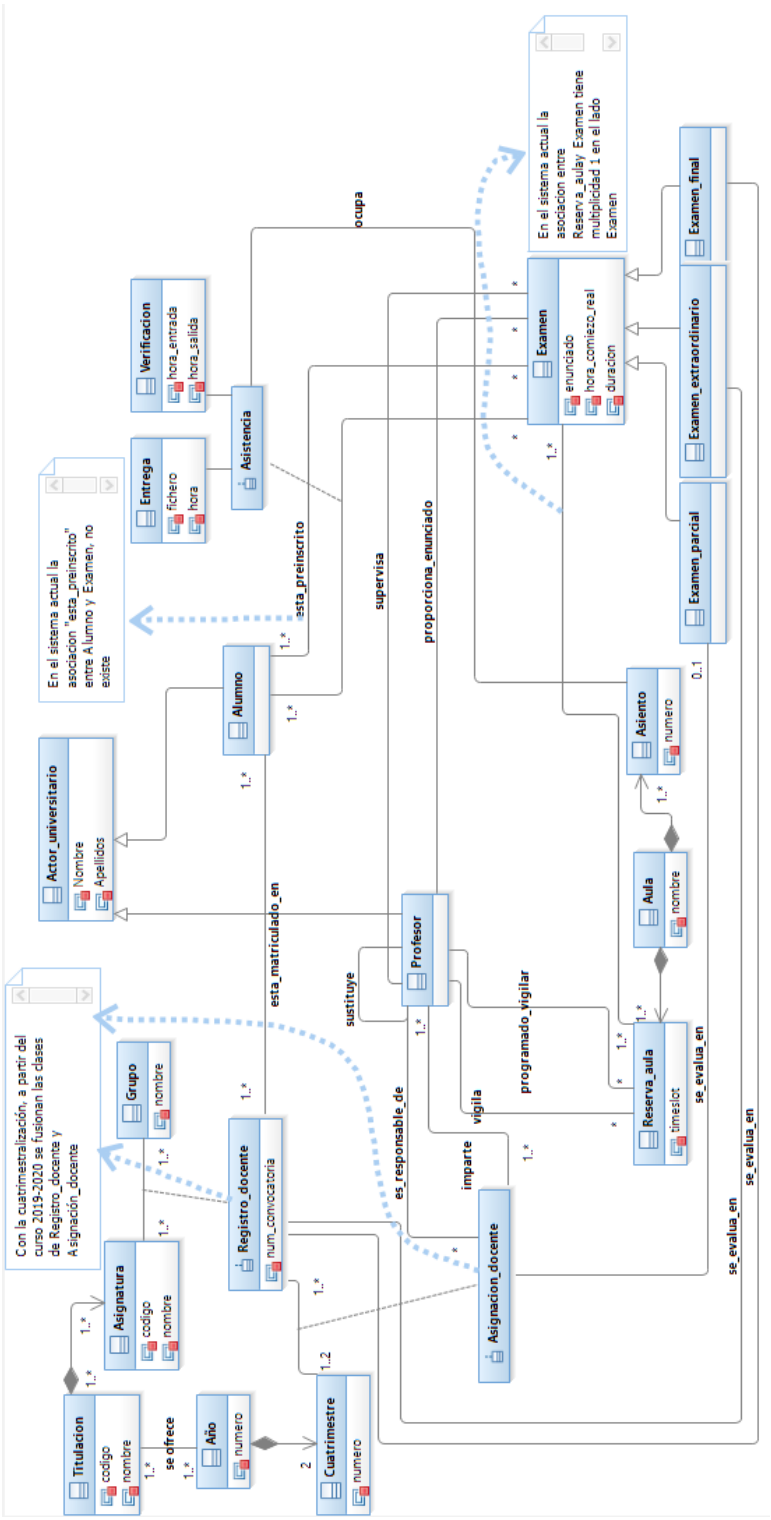

Figura 18: Modelo del dominio

4.3.1 Restricciones

Tras haber diseñado el modelo de dominio de la aplicación a desarrollar hay que tener en cuenta que no se pueden implantar ciertas restricciones de gran importancia. Por lo tanto, indicaremos a continuación dichas restricciones.

1. En una asignatura coordinada, el examen de todos los grupos es común.
2. Si un profesor es responsable de una asignación docente, supervisa el examen ordinario asociado a dicha asignación.
3. El profesor de guardia que se encarga de supervisar un examen parcial tendrá que ser el propio profesor que imparta dicha asignatura, o bien un sustituto si este no pudiese asistir o está en otro aula.
4. Quien se encarga de supervisar un examen final o extraordinario es el profesor responsable de una de las asignaciones docentes de dicho registro correspondiente, es decir, el responsable en el primer cuatrimestre , o bien el responsable en el segundo cuatrimestre. No obstante, si el responsable ha sido el mismo en ambos cuatrimestres, sería este quien supervisase. Si por algún casual uno de estos no pudiese asistir, lo supervisaría un sustituto.
5. Si un alumno se presenta a un examen, su asistencia está vinculada a un asiento del aula dónde se va a realizar. El "timeslot" asociado a dicha reserva, se comprende entre la verificación de la hora de entrada y de salida.
6. Si un examen tiene varias reservas de aula, el "timeslot" es igual para todas ellas.
7. Si un alumno se preinscribe o se presenta a un examen, está matriculado en la asignatura (registro docente) asociada a este.
8. En el modelo estático, si un alumno se quiere presentar a un examen será necesario que dicho alumno se haya preinscrito a un examen, siempre y cuando este esté matriculado en la asignatura. Sin embargo, esta asociación no existe en el sistema actual, aunque en nuestra aplicación vamos a tratar ambas: con y sin preinscripción.
9. Si un alumno está matriculado en dos registros docentes idénticos, es decir, en la misma asignatura, significa que son de distintos años.
10. Si un profesor imparte dos registros docentes idénticos significa que son de años distintos.
11. Una asignatura se ofrece un año dado si pertenece a alguna de las titulaciones de dicho año.
12. Si el número de aulas reservadas para un examen es 1, el profesor que vigila dicho aula es el responsable de la asignación docente, y, por lo tanto, supervisa el examen.

4.4 Tipos de actores o usuarios

En nuestra aplicación vamos a tratar con varios tipos de actores o usuarios, que tendrán sus respectivas funcionalidades. En concreto existen: los profesores, encargados de todo lo relacionado con los exámenes, es decir, el que finalmente decide qué exámenes se hacen; los profesores de guardia, que realizarán funciones sobre el día del examen, así como indicar alguna incidencia producida en el examen; y los profesores responsables de un grupo, quienes se encargan de comprobar que todos los exámenes hayan comenzado a tiempo y de resolver las dudas a los alumnos, entre otras funcionalidades. No obstante, cabe destacar que el profesor responsable de un grupo puede ser a su vez el profesor, por lo que las funcionalidades de dicho profesor se fusionarían con las del profesor.

Además de estos actores, también tienen especial importancia los alumnos y los administradores. Estos últimos se distribuyen en dos más: administrador del sistema y administrador de exámenes, donde el primero se encargará de comprobar que todo el sistema funcione de manera adecuada, y el segundo de la parte de decidir qué día y en qué aula se realizan los exámenes. Todos ellos emplearán la aplicación de distinta manera. Aunque en el caso de los alumnos no sea exactamente así, ya que estos no van a usar la aplicación directamente, es necesario que interactúen con ella aún no siendo de forma implícita, como en el caso de pasar la tarjeta universitaria para poder saber en qué puesto están asignados y para salir del examen, por ejemplo.

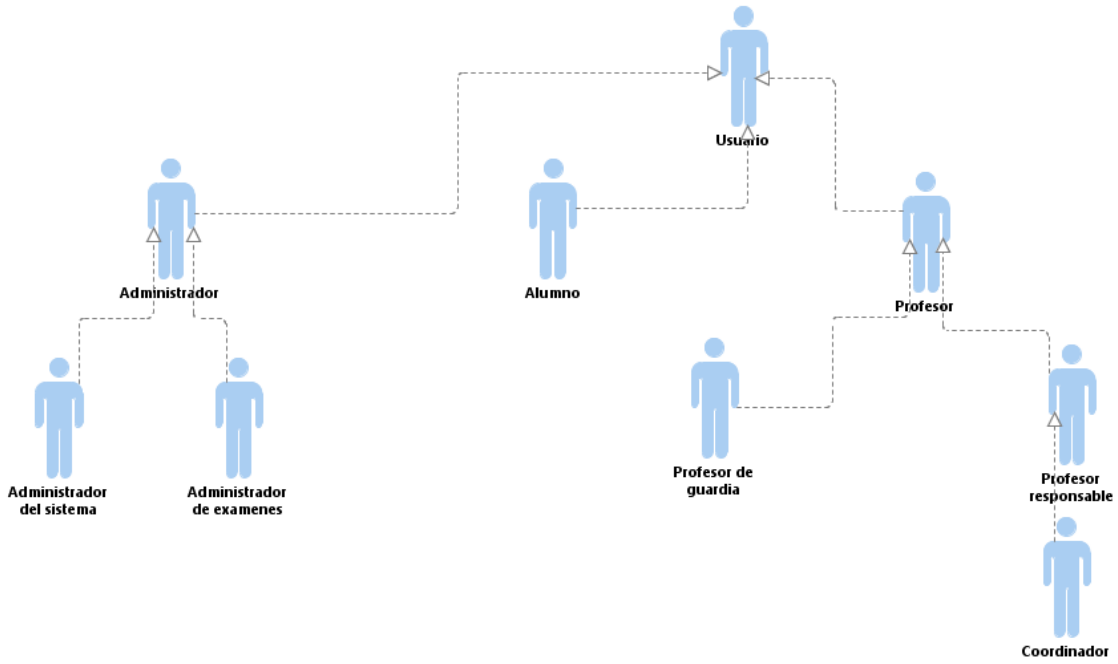

Figura 19: Tipos de usuarios

4.5 Casos de uso

Para ver con más detalle las funcionalidades que tienen cada uno de los distintos actores en la aplicación hemos empleado diagramas de casos de uso [16]. Dado que hay varios tipos de actores como ya hemos comentado en el apartado anterior, cada tipo de actor tendrá funciones específicas que podrán realizar en la aplicación. A continuación, se muestran dichos diagramas [4].

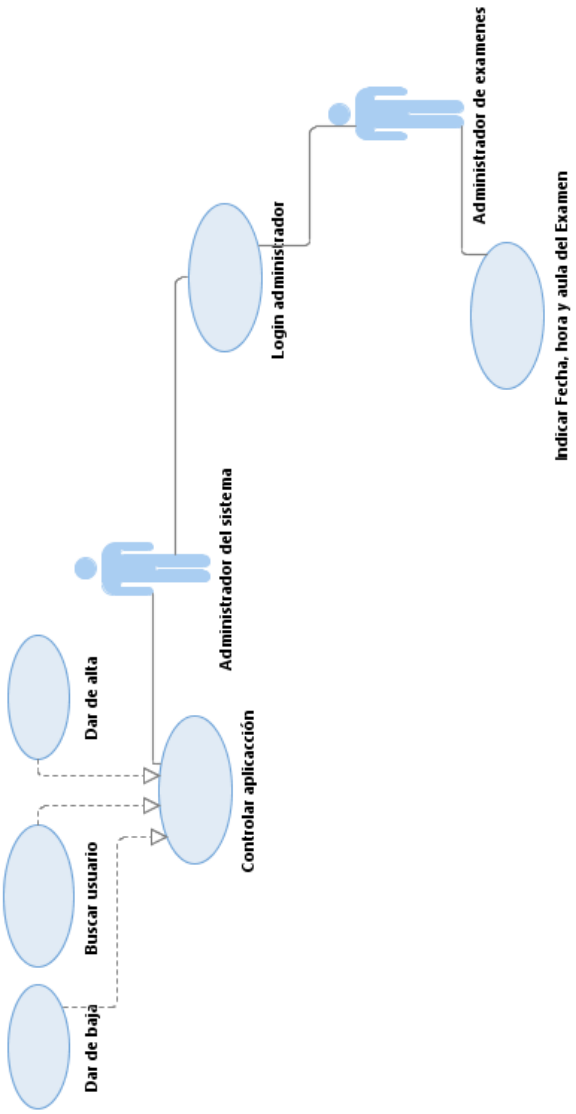

Figura 20: Casos de uso: Administrador del sistema y administrador de exámenes

[4]Todos los diagramas realizados en este apartado y en el siguiente se han hecho mediante la herramienta proporcionada por IBM Rational Software Architecture Designer [17]

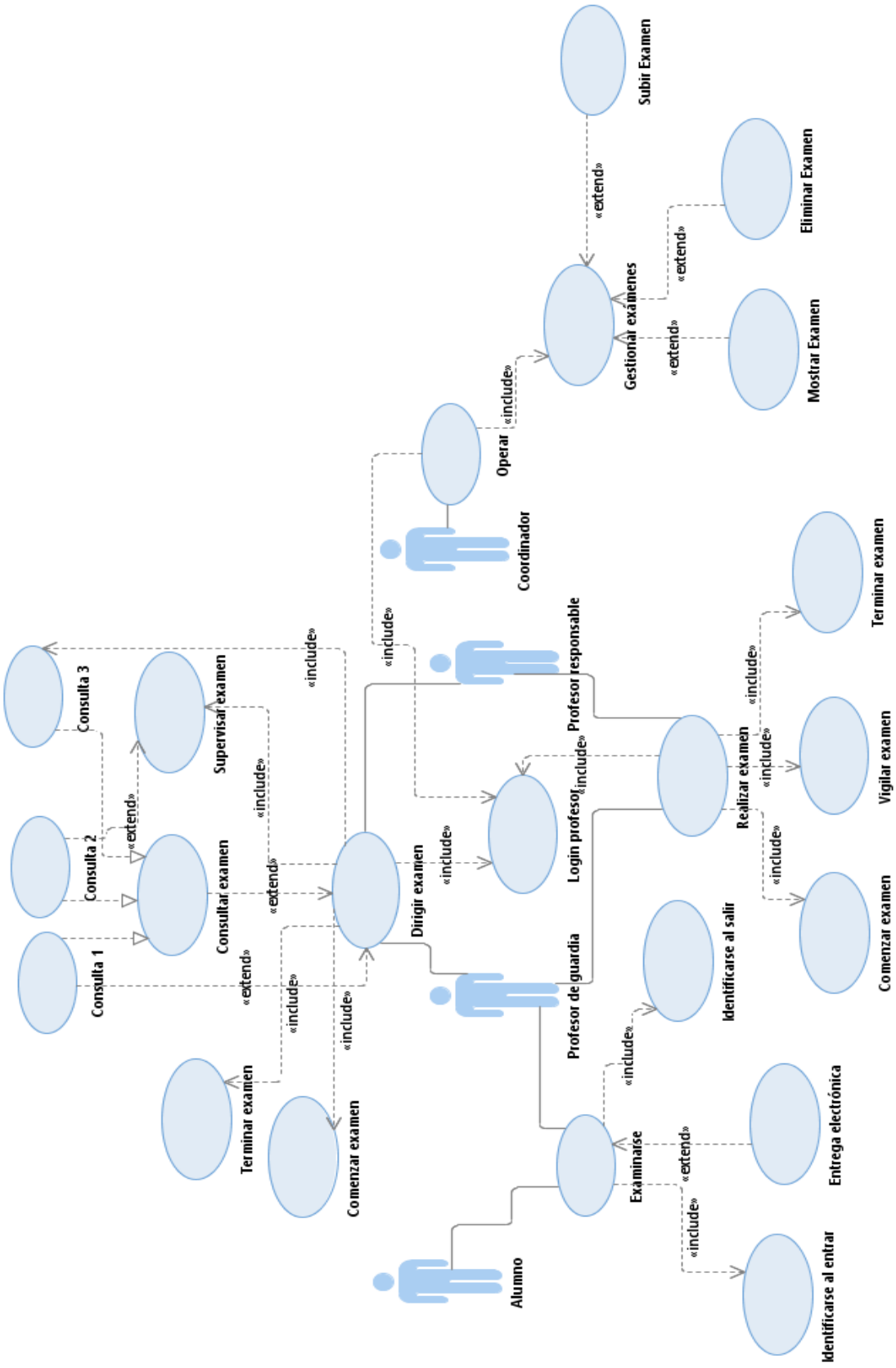

Figura 21: Casos de uso: Alumno, profesor de guardia, profesor responsable y profesor

4.5.1 Diagramas de secuencia

La ejecución de cada actor sobre la aplicación se puede ver con más detalle en los siguientes diagramas de secuencia [18]. Estos diagramas tienen especial relación con los casos de uso especificados en el apartado anterior, ya que muestran con un nivel más de detalle los pasos que realiza un actor en su caso de uso correspondiente.

Los diagramas que se muestran en las páginas siguientes están realizados con UML 2.0 [19] y representan la parte de análisis, no de diseño, basándose en lo que hacía Jacobson en su "use-case driven approach" [20]. Debido a esto están simplificados, ya que se muestra de forma general los pasos que se ejecutan internamente en la aplicación cuando un profesor añade o elimina uno o varios exámenes, por ejemplo. No obstante, para llegar a este último paso, hay un proceso de Presentación, Negocio e Integración, es decir, una arquitectura basada en las tres capas [21] para obtener, comprobar e incluir los datos correspondientes a la base de datos. Por ejemplo, en el caso de uso de Añadir examen los pasos que se siguen en las distintas capas son:

1. En la capa de Presentación lo único que habría que hacer sería extraer los datos, en el caso de añadir un examen sería coger los nombres de los modelos de exámenes que ha insertado el profesor correspondiente. Si todos los datos tienen nombres correctos los almacena y los pasa a la capa de negocio.

2. En la capa de Negocio se comprueba que no existan esos exámenes en la tabla de la BBDD de dicha asignatura. Una vez comprobado que todo es correcto los envía a Integración.

3. La capa de Integración se encarga de añadir todos los datos a la base de datos.

Por consiguiente, las dos primeras figuras corresponden al caso de uso de login tanto de los profesores como de los administradores. Tras estas aparecen los casos de uso de ambos administradores con sus correspondientes casos particulares. Seguidamente, aparecen el resto de figuras como sigue: En primer lugar, aparecen los diagramas de un profesor con sus respectivas tareas, así como subir, eliminar y/o mostrar exámenes. En segundo lugar, las acciones de un profesor de guardia sobre la aplicación en el día del examen. En tercer lugar, se puede observar las acciones que tiene que realizar un alumno que implican la aplicación cuando se realiza un examen. En cuarto y último lugar, se muestran las acciones de un profesor responsable durante las horas correspondientes a la duración del examen. Aunque se haya dividido así, se puede observar que en los distintos diagramas actúan varios usuarios, ya que los diagramas se hacen por casos de uso y varios actores pueden interactuar en un mismo caso.

A continuación, se muestran dichos diagramas para visualizar los procesos de cada usuario.

Figura 22: Diagrama de secuencia caso de uso "Login" de los administradores

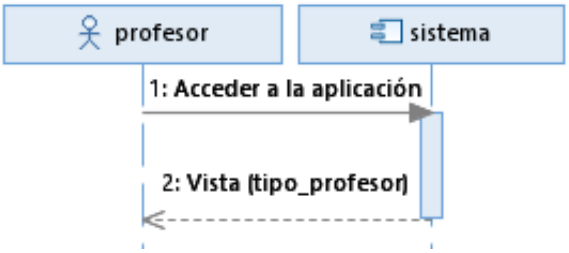

Figura 23: Diagrama de secuencia caso de uso "Login" de los profesores

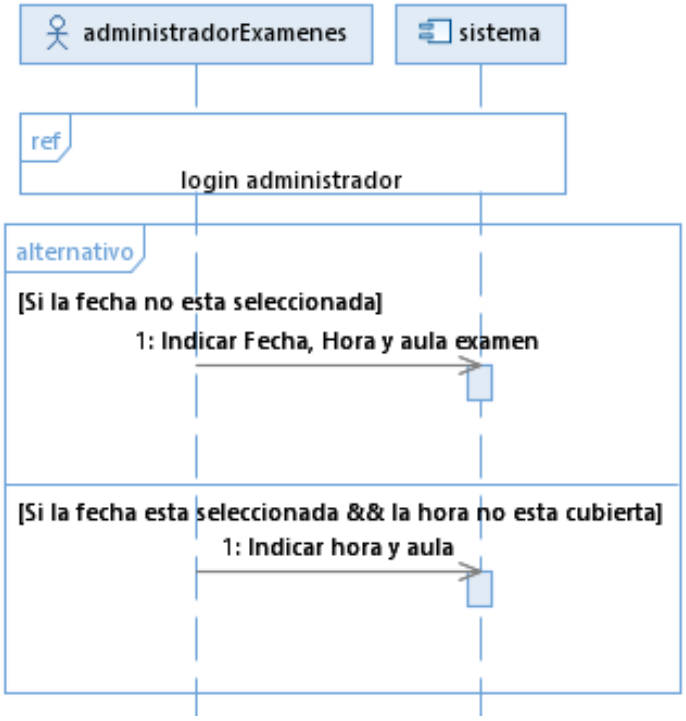

Figura 24: Diagrama de secuencia caso de uso "Indicar fecha, hora y aula examen" del administrador de exámenes

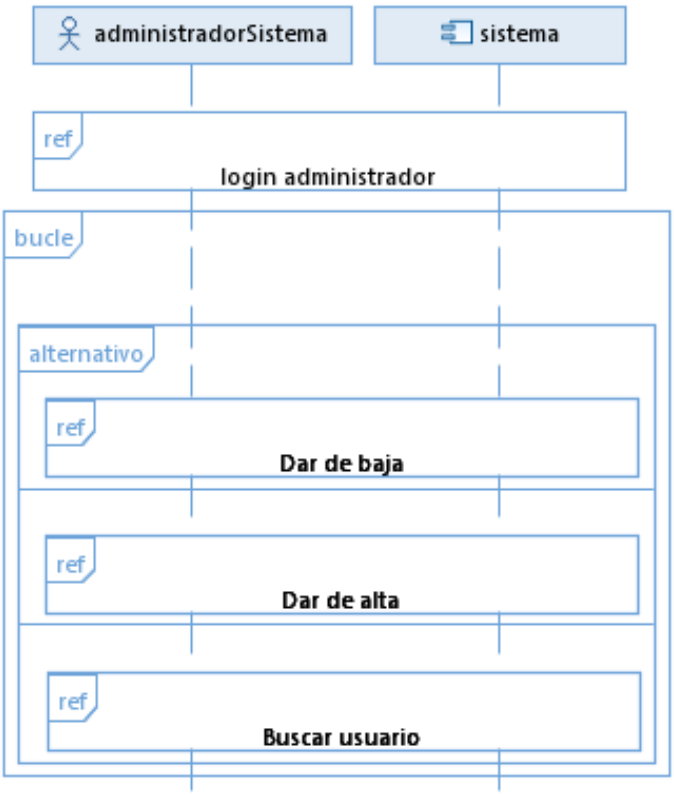

Figura 25: Diagrama de secuencia caso de uso "Controlar Aplicación" del administrador de sistema

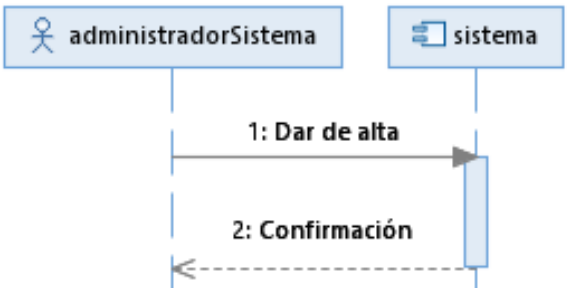

Figura 26: Diagrama de secuencia caso de uso "Dar de alta" del administrador de sistema

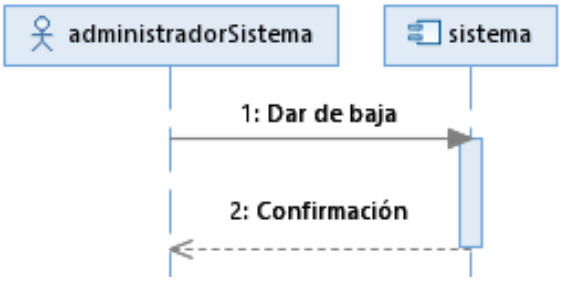

Figura 27: Diagrama de secuencia caso de uso "Dar de baja" del administrador de sistema

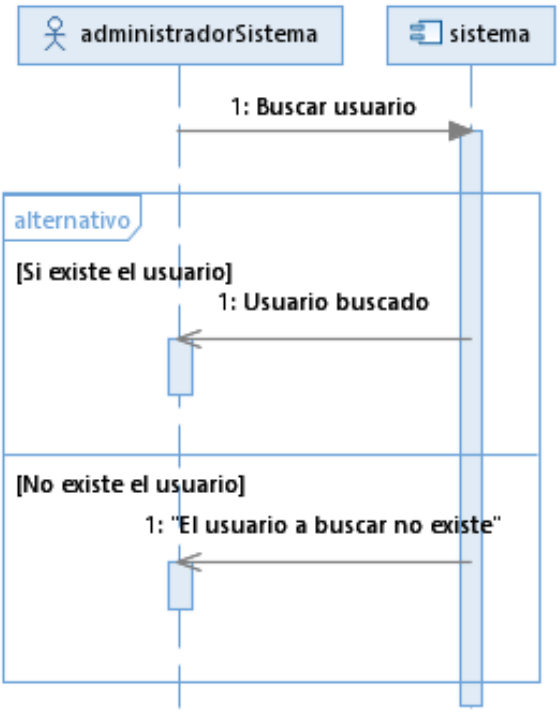

Figura 28: Diagrama de secuencia caso de uso "Buscar usuario" del administrador de sistema

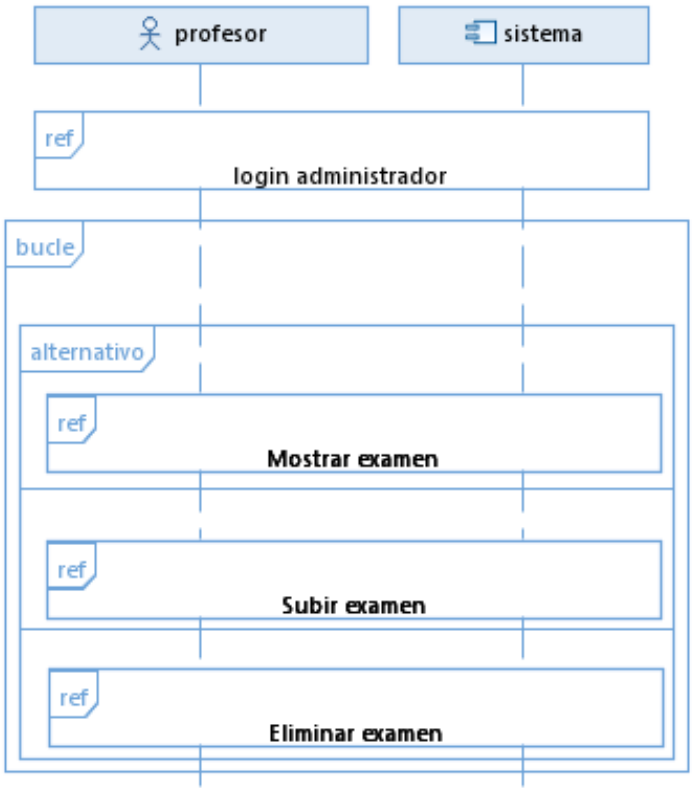

Figura 29: Diagrama de secuencia caso de uso "Operar" del profesor

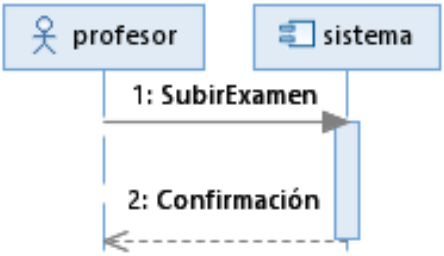

Figura 30: Diagrama de secuencia caso de uso "Subir exámenes" del profesor

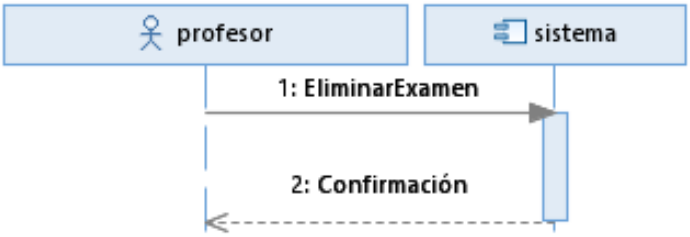

Figura 31: Diagrama de secuencia caso de uso "Eliminar exámenes" del profesor

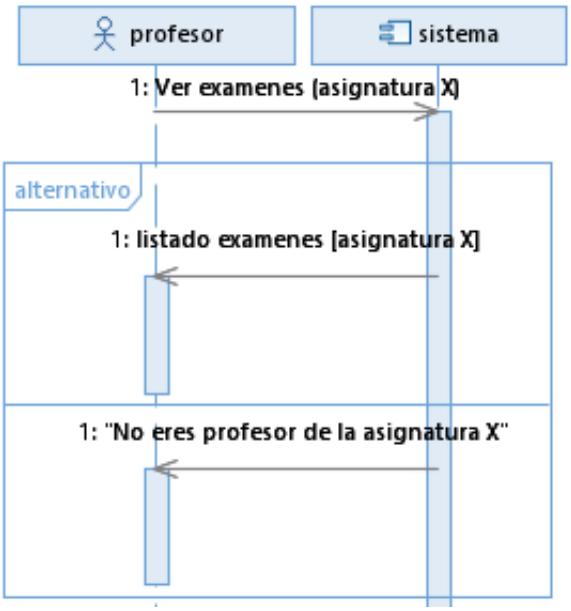

Figura 32: Diagrama de secuencia caso de uso “Mostrar exámenes” del profesor y profesor responsable

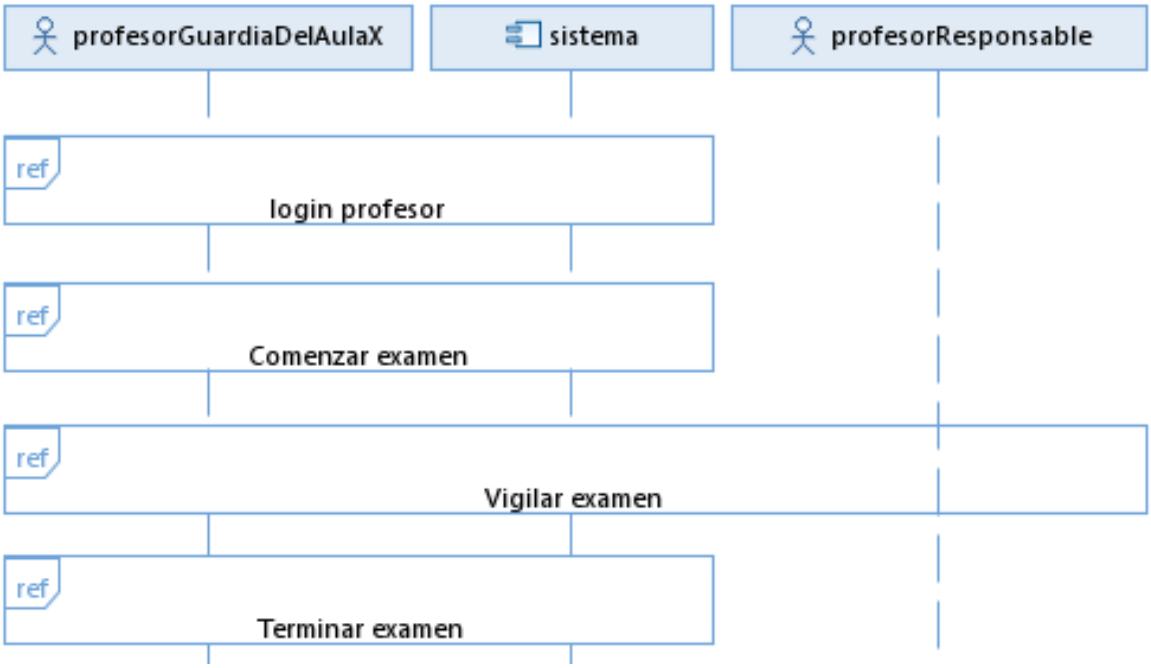

Figura 33: Diagrama de secuencia global de los casos de uso del profesor de guardia y profesor responsable

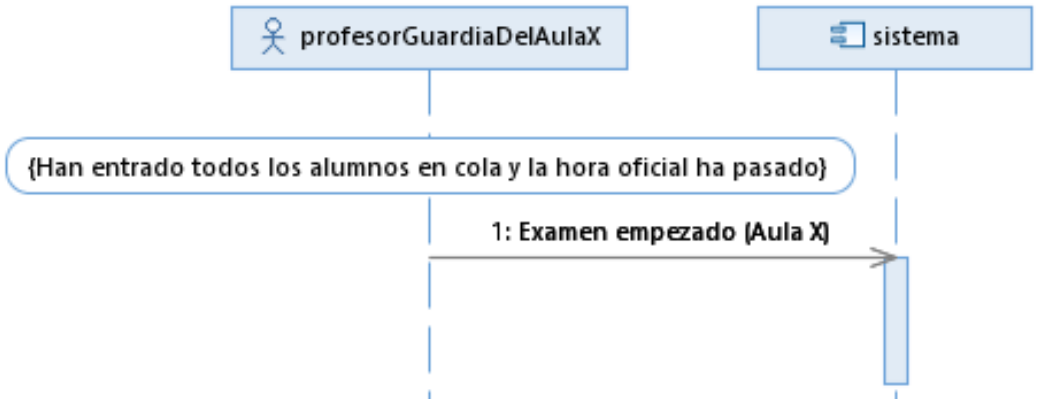

Figura 34: Diagrama de secuencia caso de uso "Comenzar examen" del profesor de guardia

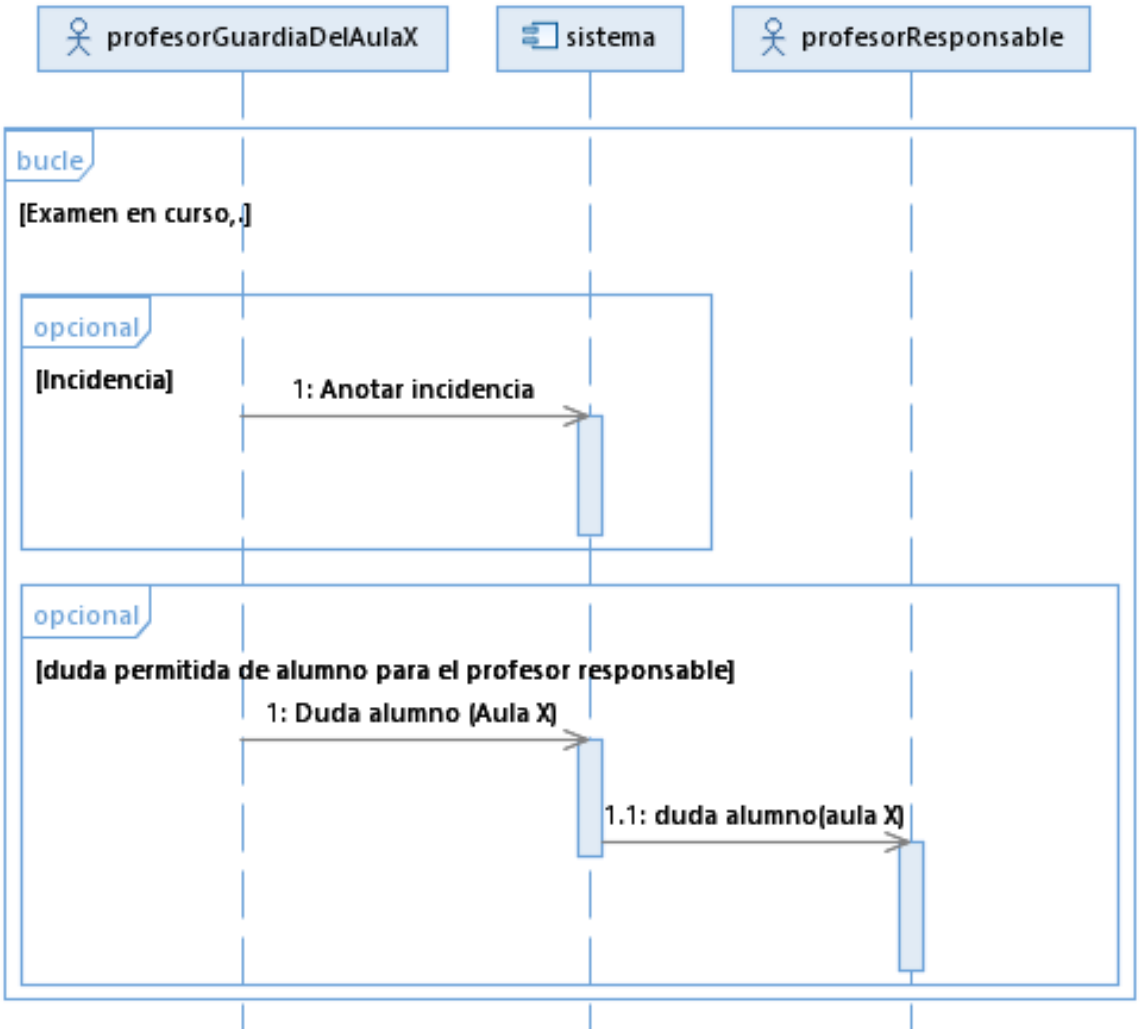

Figura 35: Diagrama de secuencia caso de uso "Vigilar examen" del profesor de guardia

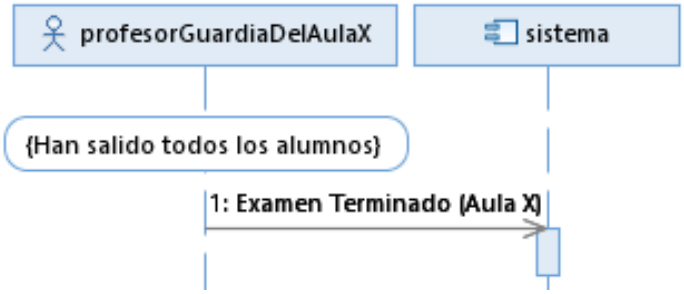

Figura 36: Diagrama de secuencia caso de uso "Terminar examen" del profesor de guardia y el profesor responsable

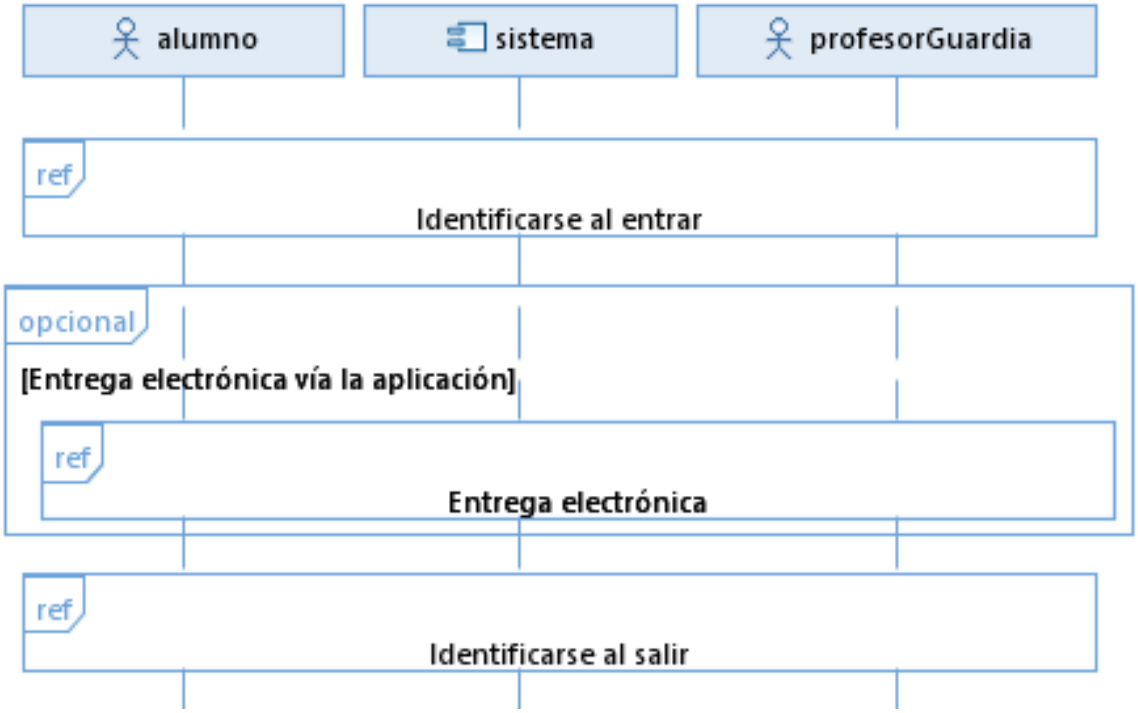

Figura 37: Diagrama de secuencia global de los casos de uso del alumno

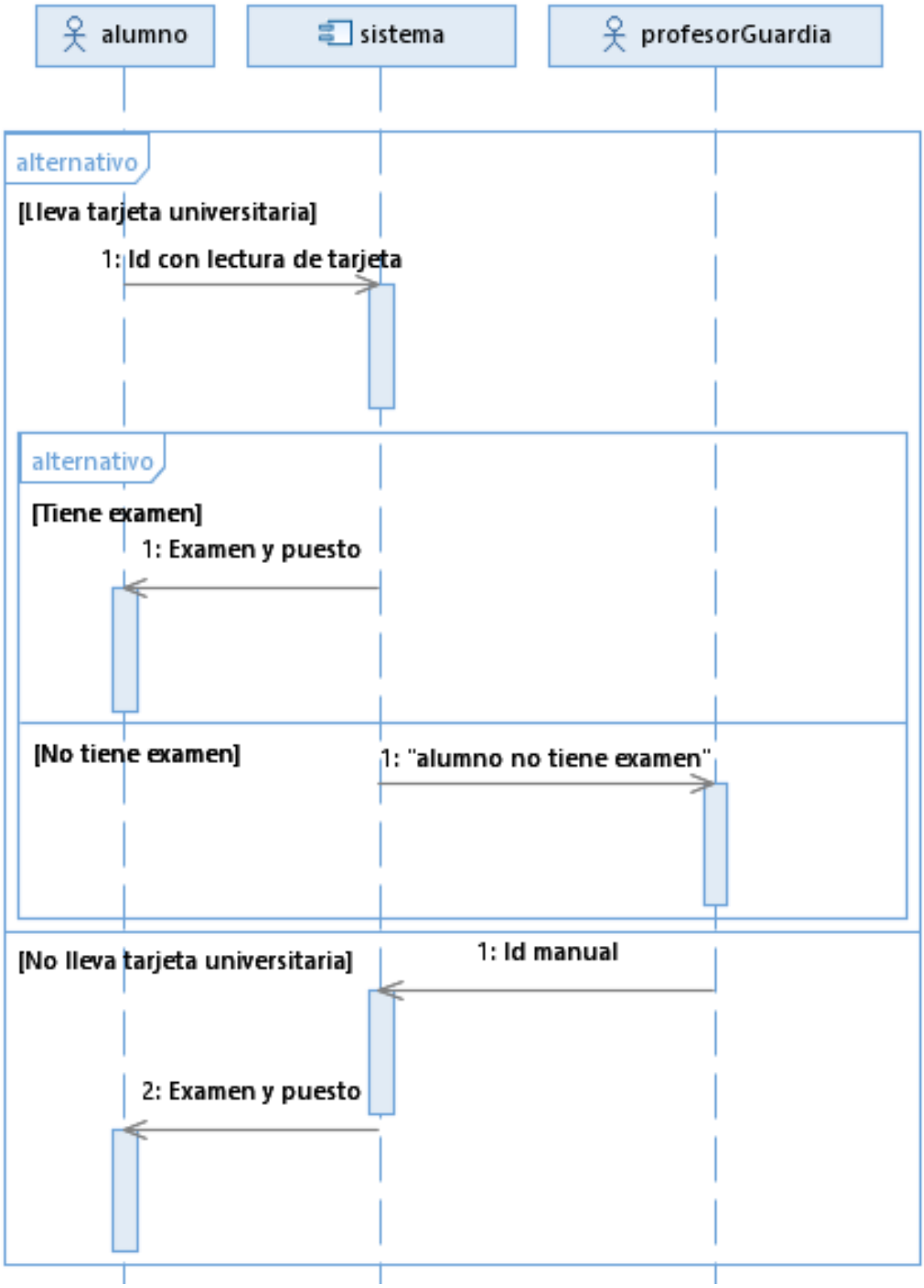

Figura 38: Diagrama de secuencia caso de uso “Identificarse al entrar” del alumno

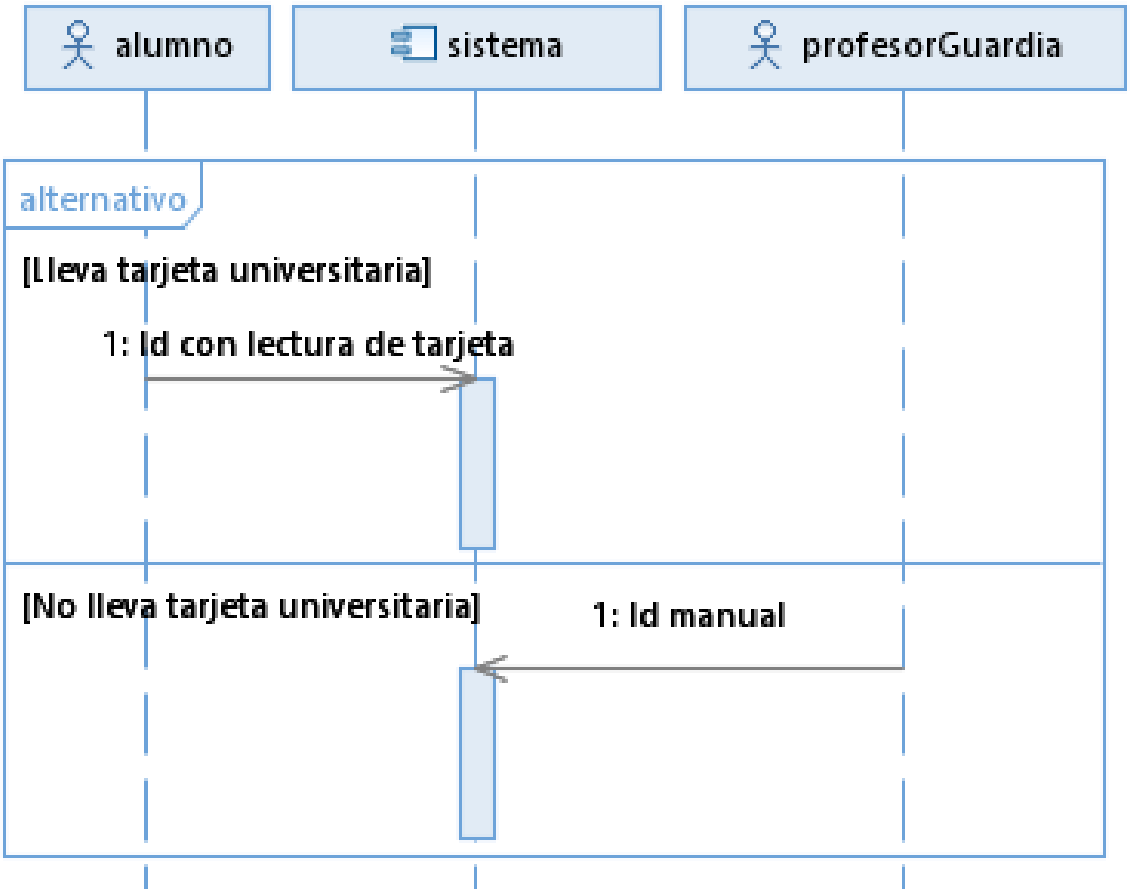

Figura 39: Diagrama de secuencia caso de uso “Identificarse al salir” del alumno

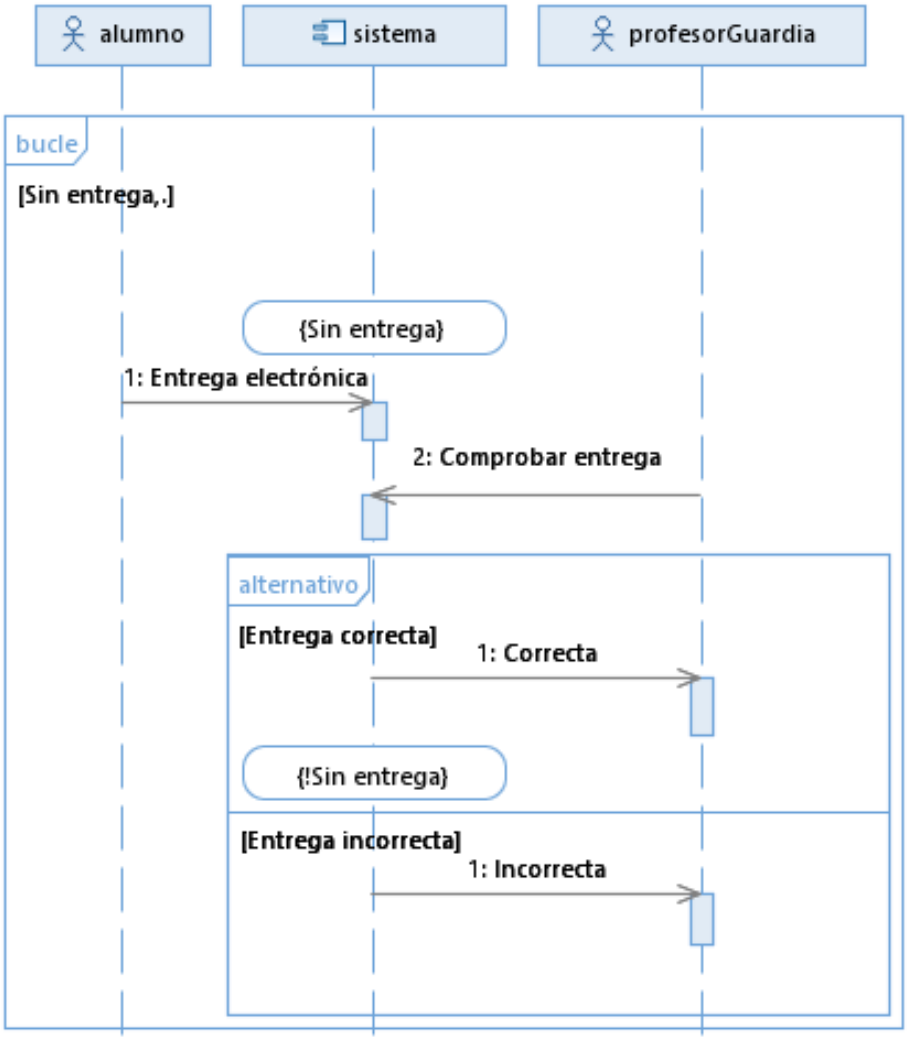

Figura 40: Diagrama de secuencia caso de uso “Entrega electrónica” del alumno

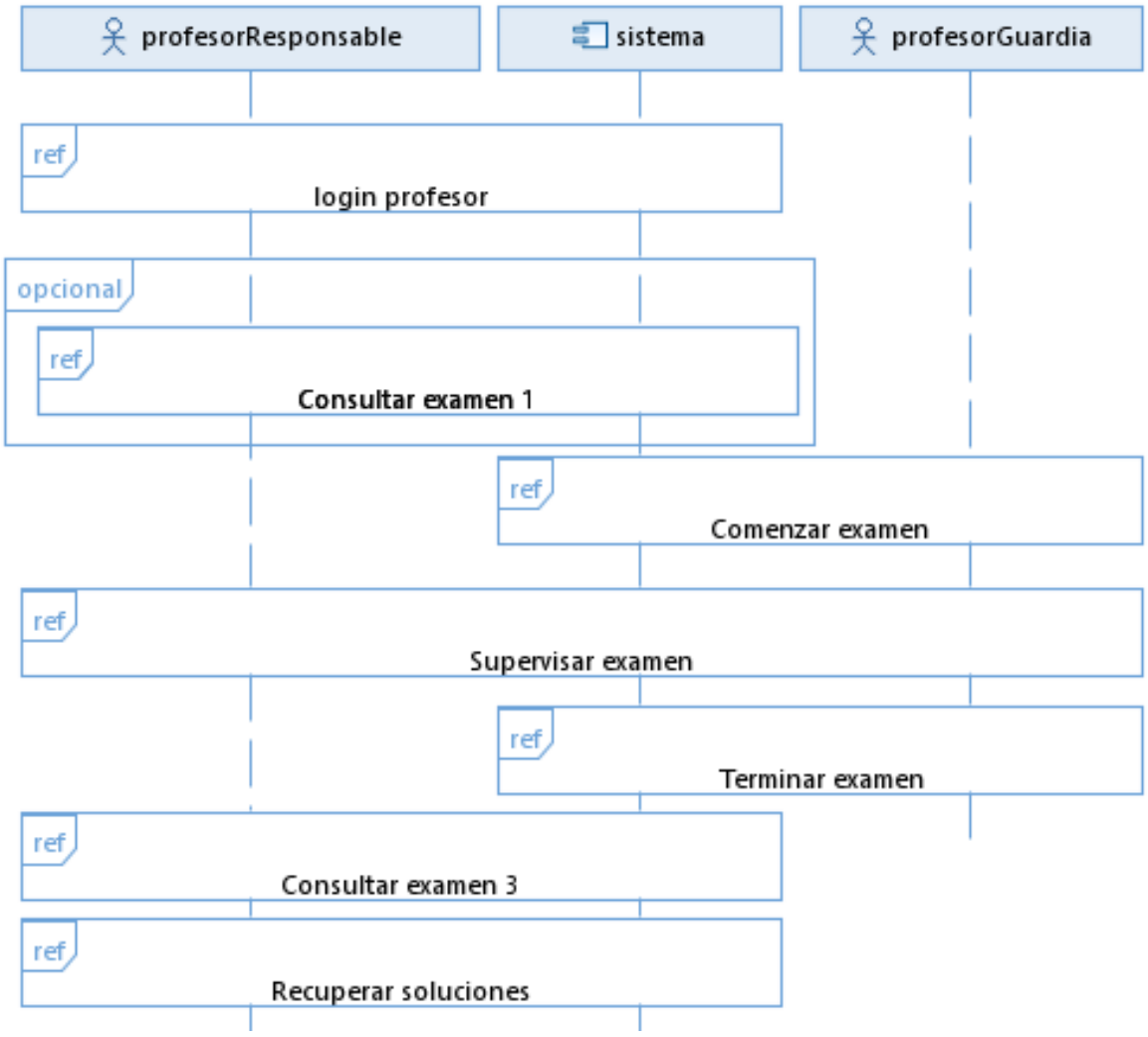

Figura 41: Diagrama de secuencia global de los casos de uso del profesor responsable

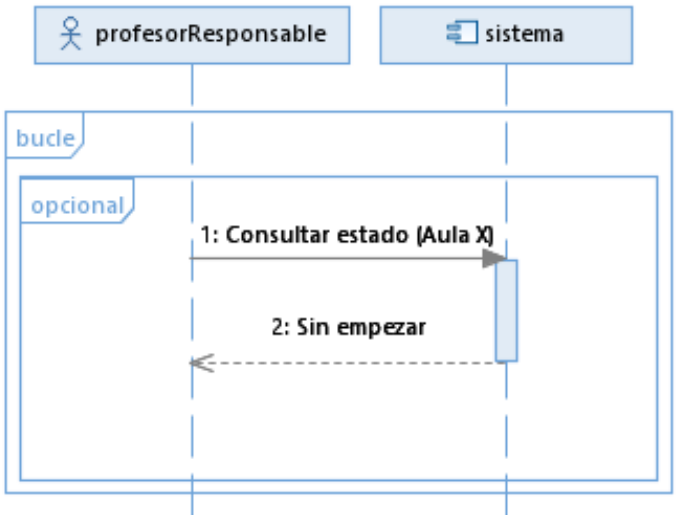

Figura 42: Diagrama de secuencia caso de uso "Consultar examen1" del profesor responsable

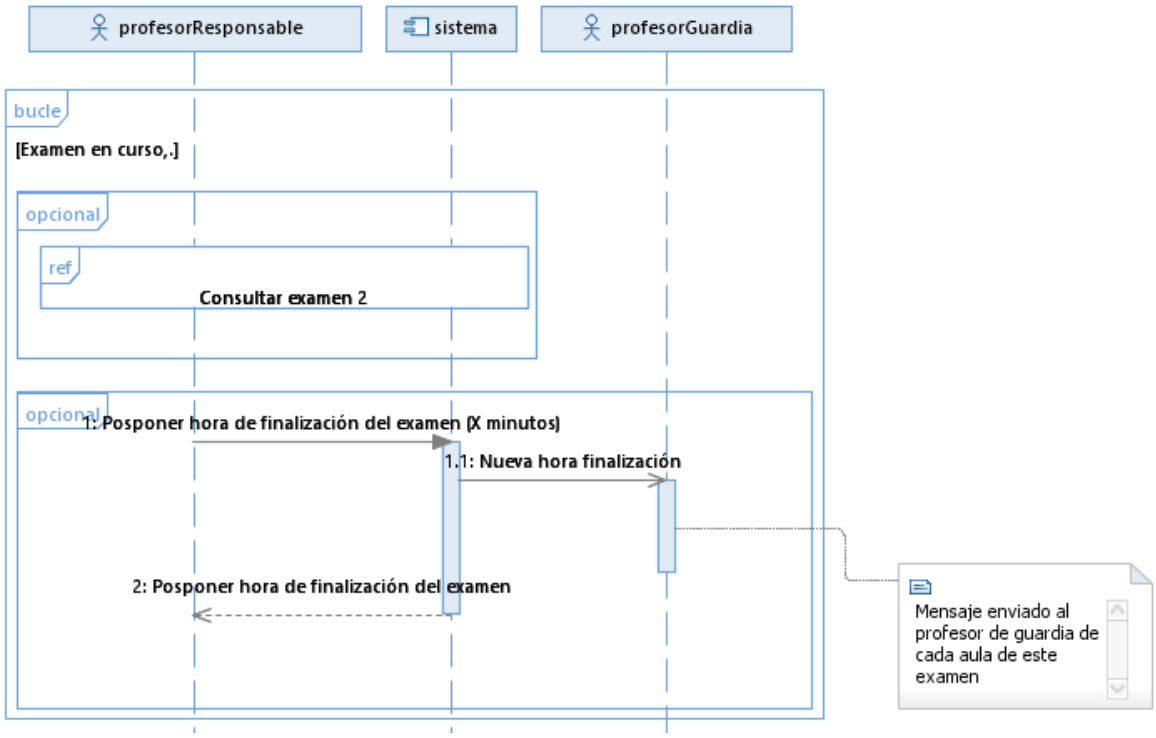

Figura 43: Diagrama de secuencia caso de uso "Supervisar examen" del profesor responsable

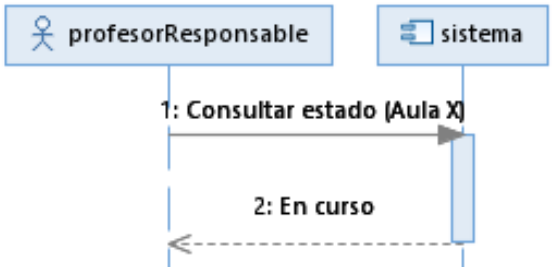

Figura 44: Diagrama de secuencia caso de uso "Consultar examen2" del profesor responsable

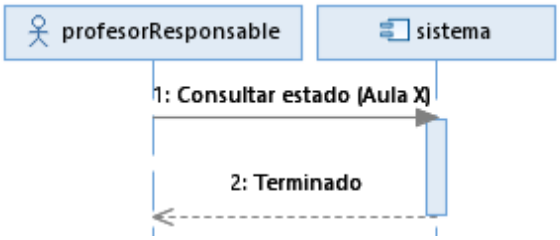

Figura 45: Diagrama de secuencia caso de uso "Consultar examen3" del profesor responsable

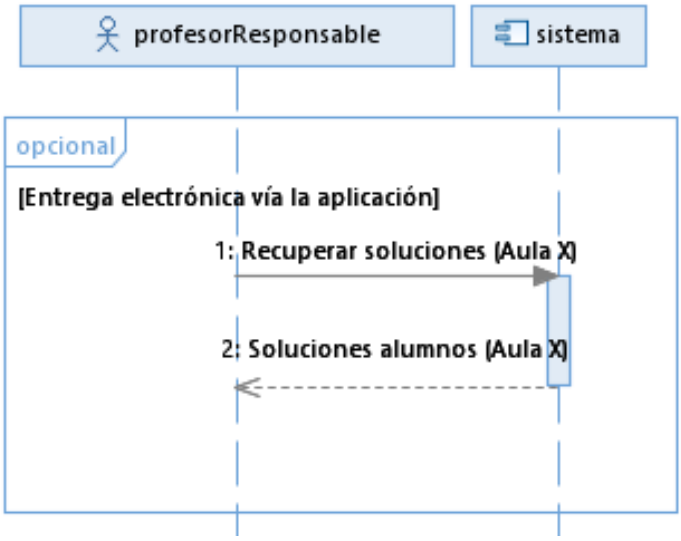

Figura 46: Diagrama de secuencia caso de uso "Recuperar soluciones" del profesor responsable

5 Verificación y validación

La verificación consiste en comprobar que el software cumple los requisitos funcionales y no funcionales de su especificación, mientras que la validación comprueba que el software cumple las expectativas que el cliente espera. Una de las técnicas más empleadas en V&V es el testing[22], empleado para evaluar la calidad del software e identificar los problemas o mejoras que puede haber.

La verificación se hará por medio de pruebas en un entorno. Con esto podremos observar si la implementación es la adecuada y no da fallos al intentar conectarse con la base de datos, o al realizar la asignación aleatoria de los puestos, entre otros.

Para validar todos los requisitos propuestos anteriormente se tendrán que hacer pruebas y ver que todo funciona correctamente. Por ello, tendremos que diseñar una interfaz de usuario y usar una base de datos de prueba, ya que no tenemos el acceso a los datos de la universidad.

En este punto es necesario tratar varios aspectos sobre el *testing*, ya que será una parte importante a la hora de comprobar que todo está realizado correctamente.

5.1 Unidad-Integración-Sistema

Para llevar a cabo la fase de *testing* será necesario realizar un proceso que incluye pruebas de unidad, de integración y de sistema, ya que así aborda todas las partes del software.

- Pruebas de unidad[23]: son aquellas que se hacen para comprobar que el código es correcto, es decir que una función concreta devuelve el resultado esperado. Para estas pruebas se podrá utilizar Mocha[24], que es un framework de Node.js.
- Pruebas de integración[25]: se llevan a cabo tras haber realizado todas las pruebas de unidad, de forma que se verifique que el software funciona correctamente en conjunto. Al igual que para las pruebas de unidad, estas pruebas también se podrán realizar con Mocha.
- Pruebas de sistema[26]: se realizan después de haber ejecutado las pruebas de unidad e integración, y el objetivo principal es probar que todo el software funcione correctamente en su entorno y en otros, de modo que pueda ejecutarse en distintas plataformas. En este caso, se podrá emplear la Prueba de Aceptación[27].

Dado que el proceso de *testing* es iterativo, cada vez que surja un fallo o alguna modificación en cada una de las pruebas se volverán a analizar de nuevo para que, al finalizar, todo esté correcto y bien *testeado*.

5.2 Caja negra-Caja blanca

Cabe decir que hay dos metodologías o estrategias que se emplean en el *testing*: caja negra y caja blanca. La primera consta en que dada una entrada de datos devuelva una salida, cuyo contexto viene dado por los requisitos. Aquí se comprueba si esta es la esperada o no. Es decir, no se tiene consciencia del código, solo se comprueba que dado unos requisitos la salida es la correcta. En cambio, la segunda se realiza con conocimiento del código y de su estructura, por lo que no será posible realizarla hasta que esté la parte del código que se quiere probar. Una de las formas de hacer este tipo de pruebas es teniendo varias sentencias if, case, while, etc. para ir comprobando partes del código y ver que se ejecuta correctamente cada bloque.

Sin embargo, estas pruebas han sido imposibles de realizar debido al tiempo que suponen y el tiempo empleado para estudiar correctamente estas herramientas. No obstante, a continuación explicamos qué tendríamos que hacer para realizar dichas pruebas en nuestra aplicación:

Por un lado, para los requisitos del sistema en los laboratorios, haremos una simulación de un examen y observaremos los fallos o no que ocasione.

Por otro lado, para los requisitos del sistema en el aula nos bastará con ver que la interfaz hace las funciones que se piden. Es decir, que se muestre la imagen del alumno en su puesto correctamente, su nombre y sus apellidos, la asignatura del examen que está haciendo y por último la opción de entrega.

En cuanto a los requisitos del usuario, se pueden validar por medio del lector de tarjetas, ya que, si no se tiene acceso a la universidad por medio de la tarjeta, no podrá realizar el examen.

Por último, para los requisitos de los exámenes se deberá hacer varias pruebas con la base de datos, subiendo varios exámenes y viendo que se han guardado correctamente. Además, se comprobará que, una vez subido los exámenes y conociendo los asistentes al mismo, si corresponde con el modelo estático, se realizará la disposición de las aulas y la asignación de los puestos de cada alumno. En el modelo dinámico, las disposición se hará en el momento del examen. Todo esto se realizará mediante pruebas, observando que todo funcione correctamente.

6 Elección de las tecnologías

6.1 Lenguajes de programación

Una vez considerados los conocimientos, la experiencia y la habilidad de los miembros del equipo en los distintos entornos y lenguajes de programación que se podrían usar de cara a la realización de este proyecto, decidimos desarrollar la aplicación web en el entorno de Visual Studio Code, ya que era un entorno familiar para todos y soportaba todos los lenguajes que considerábamos más adecuados para la programación.

Por el lado del servidor, la aplicación corre en el entorno multiplataforma de código abierto de Node.js junto con su framework Express.js, de fácil instalación gracias al Node Package Manager (npm). Los datos usados para el funcionamiento de la aplicación se encuentran alojados en la base de datos relacional MySQL, por lo que las operaciones CRUD se realizan en el lenguaje SQL.

Por el lado del cliente, la aplicación se encuentra programada con el lenguaje de marcado HTML en su versión HTML5, el lenguaje de diseño gráfico CSS para poder dar estilo al lenguaje de marcado anteriormente citado y con JavaScript implementado como parte del navegador para las mejoras, características y dinamismo de la aplicación; adicionalmente, hacemos uso de las bibliotecas de código abierto Bootstrap (basada en CSS) y JQuery (basada en JavaScript), así como la versión Standard de DHTMLX Scheduler para la interfaz gráfica de usuario de la funcionalidad del calendario de exámenes.

6.2 Requisitos exteriores

Para garantizar el correcto funcionamiento de la aplicación web, las aulas y los laboratorios deben contar con una serie de requisitos fundamentales.

Para comenzar, es imprescindible la presencia de un lector de tarjetas para poder obtener la información de cada alumno. Sin este lector, la lectura de dicha información para poder completar la interfaz y garantizar el correcto funcionamiento de la aplicación sería costoso en tiempo y mano de obra. Yuxtapuesto a este requisito podemos mencionar la necesidad de contar con un ordenador con acceso a internet; actualmente este requisito ya se encuentra garantizado en todas las aulas de la Facultad de Informática, pero cabría señalar la importancia de que el acceso a la red no estuviese activado durante las sesiones de examen. Esto es debido a que un profesor tiene la posibilidad de elegir entre distintos modos de red, deshabilitando completamente la conexión.No obstante, para nuestra aplicación sería necesario que existiese un modo, el cual se pueda deshabilitar la conexión de red en los ordenadores de los alumnos, pero manteniéndola en los de los profesores.

Asimismo, es necesaria la presencia de una una impresora rápida para cada X alumnos, donde X es un parámetro que se tiene que elegir basado en la experiencia, por ejemplo la de la UNED. Así se podrá imprimir el examen de cada alumno de forma dinámica según estos vayan fichando con su tarjeta inteligente al entrar al aula.

7 Estudio del riesgo

En este apartado vamos a tratar el riesgo de nuestra aplicación. Para ello habrá que explicar qué es el riesgo, qué tipo de riesgos hay, cómo los vamos a afrontar y qué posibles soluciones podremos aplicar.

7.1 Qué es el riesgo y tipos de riesgo

Consideraremos el riesgo como todo lo que pueda afectar de forma negativa al desarrollo del software. Vamos a calificar dichos riesgos en función de su probabilidad de aparición y del trabajo necesario para mitigarlo. En consenso, analizaremos los pros, contras y la prioridad que deberíamos asignarles. Mediante dicho estudio, podremos medir la calidad final del producto, además de controlar los costes y los errores que nos vayan surgiendo.

En la gestión de riesgos en proyectos software existen tres tipos, riesgos del proyecto, riesgos técnicos y riesgos de negocio.[28]

- Los riesgos del proyecto son cualquier acontecimiento futuro posible que puede afectar de forma negativa o positiva a nuestro proyecto. Por ejemplo, no disponer del hardware necesario para realizar nuestra aplicación, pérdida de un miembro del equipo o incluso, cambios inesperados sobre los requisitos del proyecto.
- Los riesgos técnicos ponen en peligro la calidad resultante del software. Si estos se cumplen el proyecto es más complejo de lo estimado. Tales como que los componentes de software elegidos no trabajan adecuadamente, lo que provoca fallos en el sistema, o algoritmos inadecuados que no cumplen las restricciones de tiempo de respuesta.
- Los riesgos de negocio ponen en peligro la viabilidad del software que se construirá, como podrían ser la salida de un proyecto similar al nuestro. Si estos se cumplen el proyecto se cancelará.

7.2 Estrategias de la gestión del riesgo

La gestión de riesgos puede definirse como el proceso sistemático de identificación, análisis y respuesta a los riesgos que se presentan durante el ciclo de vida de un proyecto. Para identificar riesgos y afrontarlos debidamente en el proyecto existen dos tipos de estrategias. A continuación vamos a realizar un estudio de ambas estrategias para poder determinar la más adecuada para nuestro proyecto. [29]

- Estrategia reactiva: es una acción que se efectúa en respuesta a algo que ya ha sucedido. Se realiza a posteriori, con la información del pasado disponible. El método que sigue este tipo de estrategia se basa en cinco pasos:
  1. Buscar y analizar los riesgos.
  2. Asignar recursos por si dichos riesgos se cumplen y se convierten en problemas reales.
  3. El personal no se ocupa del riesgo hasta que se hace real.
  4. Se intenta resolver el problema.
  5. Por último, comienza la gestión de crisis.
- Estrategia pro-activa: significa prever, anticipar y planear para cambios y crisis. Se aplican antes de que el riesgo haya tenido lugar. Esta también se realiza en cinco pasos que se describen a continuación:
  1. Antes de la realización del proyecto, se estudian los riesgos conocidos y por conocer.
  2. Se estima la probabilidad de que aparezcan dichos riesgos y se cuantían los problemas que puedan surgir.
  3. Se priorizan los riesgos detectados en una lista.
  4. Se realiza el plan de gestión de riesgos, previamente definido.
  5. Se realizarán planes de contingencia si no da resultado el paso anterior.

La gestión se llevará a cabo mediante tres fases distintas del plan RSGR (Reducción, Supervisión y Gestión del Riesgo). A continuación, explicamos dichas fases:

- En la fase de reducción se intentará evitar que los riesgos se conviertan en problemas reales, y se buscarán soluciones por si esto sucediera. Esta fase también se denomina como identificación del riesgo.
- La fase de supervisión controlará si los riesgos se han hecho reales. Si esto sucede, se supervisará la efectividad de los planes de reducción de riesgos obtenidos en la fase anterior. Denominada también como análisis del riego.

- Por último, cabe citar la fase de gestión del riesgo, que tiene lugar cuando se produce alguno de los riesgos que hemos estimado. Si esto sucede, se realizará un plan para que el riesgo dado tenga el menor impacto posible sobre el desarrollo del proyecto. Puede denominarse, además, como planificación del riesgo.

7.3 Elección de estrategia

Tras realizar el estudio de las estrategias llevaremos a cabo la estrategia pro-activa, debido a que los riesgos están mas controlados y se tienen planes de contingencia por si no da resultado el plan de riesgo definido.

7.4 Identificación de riesgos

En esta sección dividiremos la identificación de riesgos que pueden surgir en nuestro proyecto en las tres categorías mencionadas en la sección 7.1. Para una mayor facilidad a la hora de analizar y priorizarlos, los enumeraremos como RX, siendo X el número correspondiente y aumentando de forma creciente:

- Riesgos del proyecto
    1. R1. Entrega tardía en la documentación o el código.
    2. R2. Abandono de un miembro del equipo.
    3. R3. Falta de coordinación o comunicación por parte de los miembros del equipo.
    4. R4. Confusión con respecto a conceptos del propio proyecto.
    5. R5. Falta de implicación por parte de algún miembro del equipo.
- Riesgos técnicos
    1. R6. Falta de conocimientos de las herramientas por parte del equipo.
    2. R7. Interfaz poco amigable.
    3. R8. Mala implementación del código.
    4. R9. Fallo de la conexión con la BBDD.
    5. R10. Mal funcionamiento de las herramientas de trabajo.
    6. R11. Añadir funcionalidades no necesarias.
    7. R12. Tiempo inesperado en la ejecución de lo implementado.
- Riesgos del producto
    1. R13. Construir un producto difícil de distribuir.

2. R14. Lanzamiento de una aplicación similar antes de la nuestra.
3. R15. Falta de recursos.
4. R16. Construir un sistema no necesitado por ausencia de validación.
5. R17. La imposibilidad de integrar con los sistemas.

Tras esta identificación procedemos a analizarlos y priorizarlos según la probabilidad y el efecto que pueden llegar a tener si estos surgen.

7.5 Análisis de riesgos

Para analizar los riesgos identificados en el apartado anterior vamos a indicar la probabilidad de que ocurra y los efectos que tendría su aparición.

Este análisis se basa en el SQAS-SEI[30], por tanto, los tipos de probabilidad que hemos usado han sido:

- Frecuente: Ocurre muy frecuentemente.
- Probable: Ocurre de forma repetida.
- Ocasional: Ocurre alguna vez.
- Remota: Poco probable pero posible.
- Improbable: Con una probabilidad de ocurrencia cercana al 0.

En cuanto al efecto, a continuación, se indican los grados de gravedad que se utilizan y sus consecuencias:

- Catastrófico: Más de 6 meses de retraso con un aumento de más del 10 % del coste original y una reducción de más del 10 % en la funcionalidad.
- Crítico: Menos de 6 meses de retraso con un aumento de menos del 10 % del coste original y una reducción de menos del 10 % en la funcionalidad.
- Serio: Menos de 3 meses de retraso en la planificación con un aumento de menos del 5 % del coste original y una reducción de menos del 5 % en la funcionalidad.
- Tolerable: Menos de 1 mes de retraso en la planificación con un aumento de menos del 2 % del coste original y una reducción de menos del 2 % en la funcionalidad.
- Insignificante: Impacto insignificante.

N. º	RIESGO	PROBABILIDAD	EFECTO
R1	Entrega tardía en la documentación del proyecto	Ocasional	Crítico
R2	Abandono de un miembro del equipo	Remota	Serio
R3	Falta de coordinación o comunicación por parte de los miembros del equipo	Probable	Serio
R4	Confusión con respecto a conceptos del propio proyecto	Ocasional	Catastrófico
R5	Falta de implicación por parte de algún miembro del equipo	Ocasional	Crítico
R6	Falta de conocimientos de las herramientas por parte del equipo	Improbable	Crítico
R7	Interfaz poco amigable	Improbable	Tolerable
R8	Mala implementación del código	Improbable	Catastrófico
R9	Fallo de la conexión de la BBDD	Ocasional	Catastrófico
R10	Mal funcionamiento de las herramientas de trabajo	Improbable	Tolerable
R11	Añadir funcionalidades no necesarias	Improbable	Insignificante
R12	Tiempo de funcionalidad inesperado	Ocasional	Serio
R13	Construir un producto difícil de distribuir	Ocasional	Tolerable
R14	Lanzamiento de una aplicación similar antes de la nuestra	Improbable	Serio
R15	Falta de recursos	Ocasional	Tolerable
R16	Construir un Sistema no necesitado	Remota	Tolerable
R17	Imposibilidad de integrar con los sistemas	Improbable	Tolerable

Figura 47: Análisis de riesgos

7.6 Priorización del riesgo

A continuación, priorizamos los riesgos más relevantes según la probabilidad y el efecto de éstos descritos en la tabla anterior. En este apartado también hemos seguido la técnica del SQAS-SEI.

	Frecuente	Probable	Ocasional	Improbable	Remota
Catastrófico			R4, R9	R8	
Crítico			R1, R5	R6	
Serio		R3	R12	R14	R2
Tolerable			R13, R15	R7, R10, R17	R16
Insignificante				R11	
LEYENDA	INTOLERABLE	ALTO	MEDIO	BAJO	TOLERABLE

Figura 48: Priorización del riesgo

Para filtrar estos riesgos utilizamos la Regla o Principio de Pareto[31], escogiendo el 20 % de los riesgos identificados. Dado que tenemos 17 riesgos, redondeamos el 20 % a los 4 riesgos mayores. En concreto, hemos elegido R4, R9, R3 y R1 que corresponden a riesgos intolerables y altos.

N. º	TIPO	RIESGO	PROBABILIDAD	EFECTO	GRAVEDAD
1	Proyecto	4. Confusión con respecto a conceptos del propio proyecto	Ocasional	Catastrófico	Intolerable
2	Técnico	9. Fallo de la conexión con la BBDD	Ocasional	Catastrófico	Intolerable
3	Proyecto	3. Falta de coordinación o comunicación por parte de los miembros del equipo	Probable	Serio	Alto
4	Proyecto	1. Entrega tardía en la documentación o el código	Ocasional	Crítico	Alto
5	Proyecto	5. Falta de implicación por parte de algún miembro del equipo	Ocasional	Crítico	Alto
6	Técnico	8. Mala implementación del código	Improbable	Catastrófico	Alto
7	Técnico	12. Tiempo inesperado en la ejecución de lo implementado	Ocasional	Serio	Medio
8	Técnico	6. Falta de conocimientos de las herramientas por parte del equipo	Improbable	Crítico	Medio
9	Producto	13. Construir un producto difícil de distribuir	Ocasional	Tolerable	Bajo
10	Producto	15. Falta de recursos	Ocasional	Tolerable	Bajo
11	Producto	14. Lanzamiento de una aplicación similar antes de la nuestra	Improbable	Serio	Bajo
12	Técnico	7. Interfaz poco amigable	Improbable	Tolerable	Tolerable
13	Técnico	10. Mal funcionamiento de las herramientas de trabajo	Improbable	Tolerable	Tolerable
14	Producto	17. Imposibilidad de integrar con los sistemas	Improbable	Tolerable	Tolerable
15	Proyecto	2. Abandono de un miembro del equipo	Remota	Serio	Tolerable
16	Producto	16. Construir un sistema no necesitado por ausencia de validación	Remota	Tolerable	Tolerable
17	Técnico	11. Añadir funcionalidades no necesarias	Improbable	Insignificante	Tolerable

Figura 49: Análisis y priorización del riesgo

7.7 Técnicas de Reducción, Supervisión y Gestión del Riesgo

Una vez identificados, analizados y priorizados todos los riesgos, vamos a llevar a cabo el Plan de Reducción, Supervisión y Gestión del Riesgo para cada uno de los riesgos seleccionados en el apartado anterior. Para ello, se realizarán tres fases:

1. **Reducción:** Lo que se va a hacer para evitar que el riesgo ocurra.
2. **Supervisión:** Lo que se va a hacer para controlar si el riesgo se ha hecho real.
3. **Gestión:** Lo que se hará si el riesgo se hace real.

A continuación, vamos a realizar dicho plan RSGR sobre los riesgos seleccionados:

R4: Confusión con respecto a conceptos del propio proyecto.

1. Reducción: Llevar una organización del equipo respecto a los conceptos exigidos.
2. Supervisión: Comprobar periódicamente que los conceptos aplicados son los que se piden.
3. Gestión: Corregir los conceptos erróneos.

R9: Fallo de la conexión con la BBDD.

1. Reducción: Crear una Base de Datos normalizada.
2. Supervisión: Comprobar en cada cambio que se conecta bien con la Base de Datos.

3. Gestión: Corregir posibles relaciones entre las tablas de la Base de Datos.

R3: Falta de coordinación o comunicación por parte de los miembros de equipo.

1. Reducción: Cada miembro del equipo expondrá sus dudas o propuestas para la organización en el proyecto.
2. Supervisión: Se hará una reunión cada semana o dos semanas exponiendo cada miembro los problemas o dudas que tengan sobre el avance del proyecto
3. Gestión: Volver a planificar y resolver los problemas de comunicación.

R1: Entrega tardía en la documentación o el código.

1. Reducción: Entregas semanales de documentación y/o código.
2. Supervisión: Se harán reuniones cada dos semanas para verificar que todo esté correctamente.
3. Gestión: Ponerse todo el equipo con la parte que se está tardando y hacerlo en el menor tiempo posible.

8 El sistema

8.1 Introducción

Nuestro sistema se organiza según el patrón MVC, uno de los patrones de arquitectura de software más comunes en el desarrollo de aplicaciones con interfaces de usuario. Este patrón divide la aplicación en tres partes interconectadas: el modelo de datos, las interfaces de usuario y la lógica de control. La lógica de control (Controlador) hace de nexo de unión entre el modelo de datos y las vistas del sistema. De esta forma, el patrón se basa en las ideas de reutilización de código y la separación de conceptos con el objetivo de facilitar el desarrollo de aplicaciones y su posterior mantenimiento.Dentro de este capítulo explicaremos los componentes, características y funcionalidades del sistema, estructurándolo desde el punto de vista de este patrón. A continuación se muestra una estructura de la relación entre el controlador, la vista y el modelo.

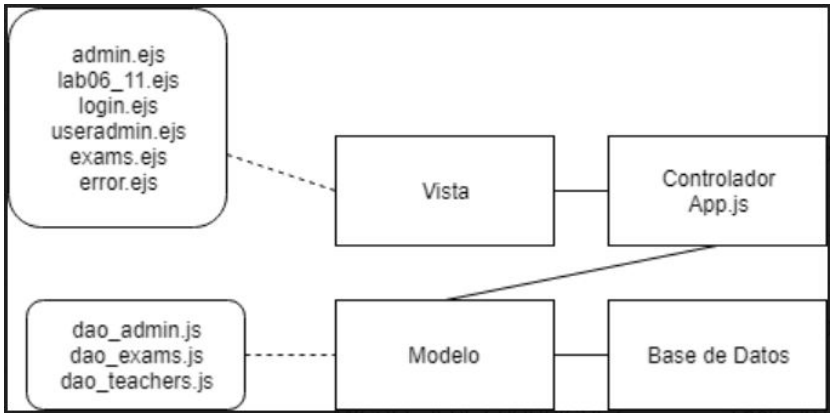

Figura 50: Estructura de la aplicación.

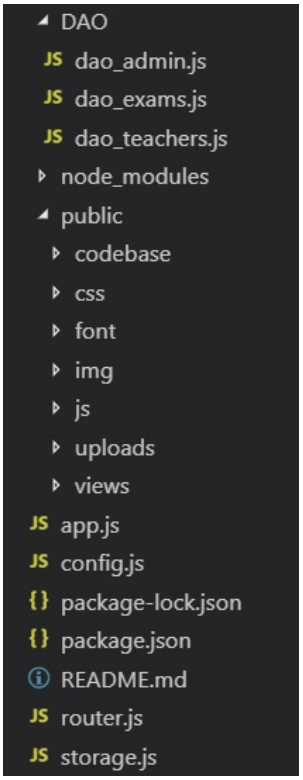

Figura 51: Árbol de organización

En la figura de la izquierda podemos encontrar una imagen con el árbol de organización de nuestro sistema. Dentro de este árbol, la parte de Modelo se encontraría dentro de la carpeta DAO, mientras que la Vista se encontraría dentro de la carpeta *public*, cuyas subcarpetas explicaremos en la correspondiente sección. El controlador, la capa que invoca peticiones al modelo y las acaba enviando a la vista, se encontraría concentrado en el archivo app.js.

Al estar desarrollada en Node.js, en este árbol también podemos ver que la aplicación contiene una carpeta llamada *node_modules* y dos los archivos en la raíz llamados *package.json* y *package-lock.json*. Estas carpetas y archivos se crean automáticamente al usar por primera vez npm, el manejador de paquetes por defecto para Node.js y parte fundamental en el desarrollo de aplicaciones en dicho entorno. Gracias a esta herramienta, el desarrollador tiene acceso a multitud de módulos de software libre creados por la comunidad para multitud de propósitos diferentes dentro del desarrollo web. De esta forma, los ficheros *package.json* y *package-lock.json* contienen un listado de los módulos que han sido descargados a través de la consola npm, mientras que en la carpeta *node_modules* encontramos el contenido de dichos módulos, necesarios para una compilación exitosa del proyecto.

8.2 Modelo del sistema

Dentro de la capa de modelo, podemos diferenciar tres tipos de DAO en función del tipo de instrucciones y consultas que se realizan desde cada uno de ellos:

- **dao_admin.js**: Este archivo js contiene las funciones relativas a la administración de los usuarios que contiene la BBDD tales como añadir, eliminar, buscar o devolver uno o una lista de usuarios.
- **dao_exams.js**: el DAO exámenes sirve de conexión con la BBDD a la hora de manipular la información referente a los exámenes, tales como su subida a la base de datos (BBDD), su eliminación de la misma o mostrar aquellos que están guardados, entre otras funcionalidades.
- **dao_teachers.js**: Este último DAO contiene las funciones necesarias para realizar las operaciones CRUD relativas al profesorado, ente las que podemos citar el control de acceso de usuarios, la consulta de sus asignaturas en curso o la consulta sobre un posible examen en marcha al iniciar la aplicación.

```
/**
 * Determina si un determinado usuario aparece en la BD con la contraseña
 * pasada como parámetro.
 *
 * Es una operación asíncrona, de modo que se llamará a la función callback
 * pasando, por un lado, el objeto Error (si se produce, o null en caso contrario)
 * y, por otro lado, un booleano indicando el resultado de la operación
 * (true => el usuario existe, false => el usuario no existe o la contraseña es incorrecta)
 * En caso de error error, el segundo parámetro de la función callback será indefinido.
 *
 * @param {string} user Identificador del usuario a buscar
 * @param {string} password Contraseña a comprobar
 * @param {function} callback Función que recibirá el objeto error y el resultado
 */
isUserCorrect(user, password, callback) {
    this.pool.getConnection((err, connection) => {
        if(err){
            callback(err, undefined);
        }
        else{
            connection.query("SELECT * FROM conectar  WHERE user = ? and pass = ?",
            [user, password],
            (err, rows) => {
                connection.release();
                if (err) { callback(err, undefined); }
                if (rows.length === 0) {
                    callback("Usuario y/o contraseña incorrectos", undefined);
                } else {
                    callback(null, rows[0]);
                }
            });
        }
    });
}
```

Figura 52: Ejemplo de una función del dao_teachers, en este caso de la función que controla el acceso a la aplicación

Dada la falta de presupuesto para adquirir los elementos necesarios para el proyecto, como pueden ser el lector de tarjetas y las impresoras, se han tenido que simular algunos procesos. Para asignar puestos a los alumnos en un aula de examen se añaden aleatoriamente a partir de la información de la base de datos y se les asigna un puesto aleatorio que tiene a su vez asignado un modelo con la restricción de que un alumno no puede estar al lado de otro con el mismo modelo. Respecto a la falta de impresoras, se ha creado una función donde para cada alumno que entra a una sala de examen, su modelo se añade a una cola de impresión.

8.3 Vistas del sistema

En cuanto a la última capa, relativa a la interfaz de usuario, las vistas del sistema se encuentran ubicadas dentro del directorio public/views, dentro del cual se pueden observar varios archivos del tipo ".ejs". Este tipo de archivos consisten en plantillas de JavaScript que permiten procesar páginas HTML en el servidor antes de enviarlas al cliente. Es uno de los motores de plantillas más rápidos cuando se usa junto a Node.js y Express.js. Como ha sido explicado anteriormente, dependiendo de si el usuario tiene el rol de administrador o el del profesor, la aplicación permite el acceso a unas vistas u otras; a continuación se enumeran y explican brevemente estas interfaces:

- **adminusers.ejs:** Esta vista solo está disponible para el administrador del sistema. En ella se muestran todos los actores universitarios con toda su información (id, Nombre, Apellidos y Rol) junto con la opción de agregar, gestionar credenciales de acceso, gestionar supervisores de exámenes y borrar usuarios.

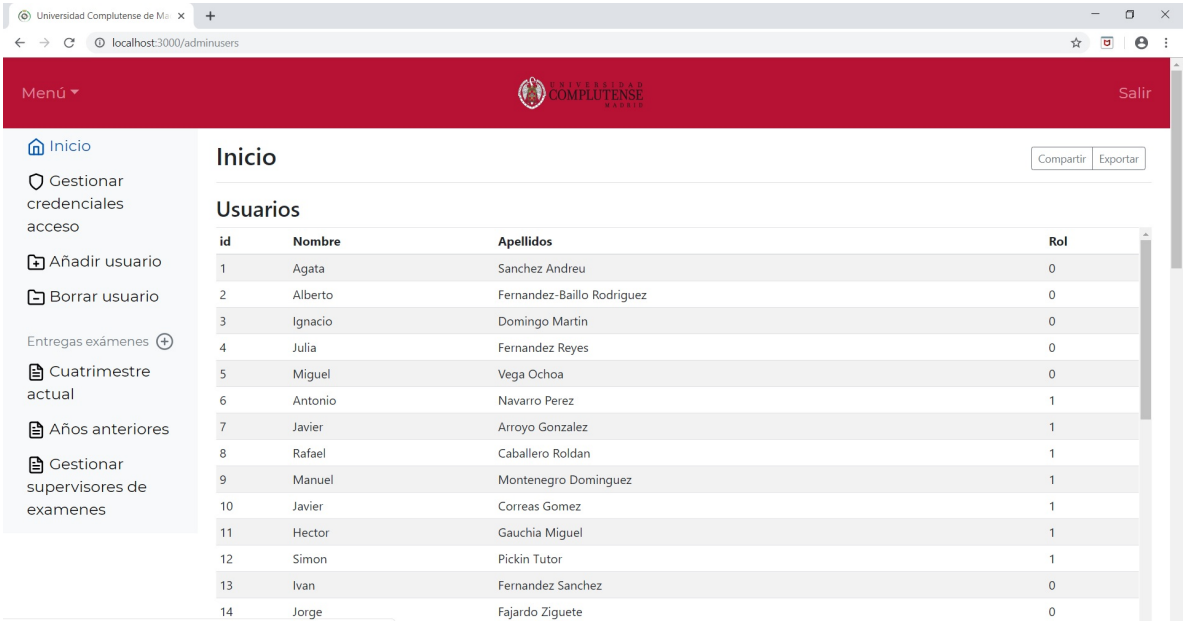

Figura 53: Interfaz de la vista de administración de usuarios.

- **admin.ejs:** Esta vista muestra un calendario para que el administrador de exámenes, único usuario que puede acceder a ella, pueda establecer los días, horas y aulas para la realización de los exámenes de los distintos grados y cursos. Esta vista ofrece también al administrador la posibilidad de exportar dicho calendario en formatos PNG y PDF.

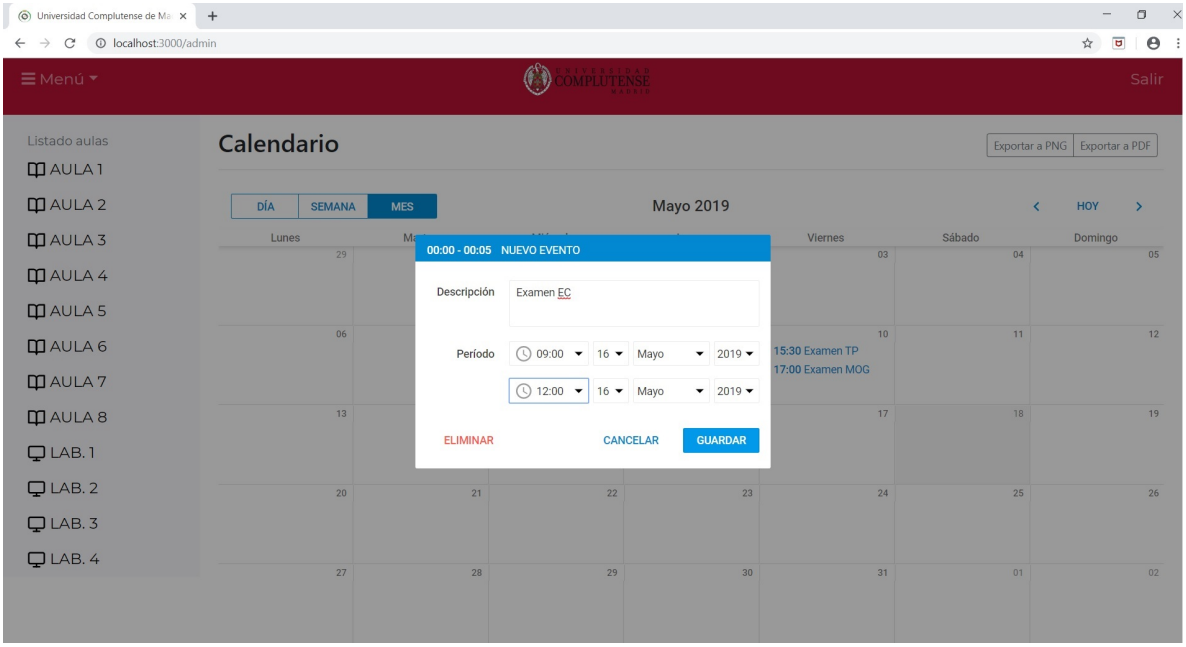

Figura 54: Interfaz de la vista de administración de exámenes. En la captura de pantalla podemos ver el momento justamente anterior a la introducción del examen en la base de datos.

- **hall.ejs:** Esta interfaz representa la distribución del aula o laboratorio en el que se está realizando el examen de la asignatura que imparte el profesor con la sesión iniciada. En la parte superior se indica el aula en el que se imparte el/los examen/es junto con el nombre de la/s asignatura/s. A continuación se encuentra, en la parte izquierda, una imagen que representa el aula con el número de filas y columnas y el número de los puestos, y, a su derecha, aparece una tabla donde cada fila corresponde a la información de cada alumno que va llegando al aula; incluye su número de asiento, su nombre completo y el modelo del examen que le ha asignado el algoritmo en función de su puesto. Además, al seleccionar una fila se despliega una ventana modal con la información del alumno que está realizando el examen en ese puesto. Dentro de ella, también se puede observar varios botones: uno con la opción de indicar la entrega del examen por parte del alumno, otro para indicar al profesor una posible situación de copia que haya incluido a ese alumno en el transcurso del examen, otro para los alumnos que han decidido marcharse sin entregar si los estatutos así lo permiten, y un último para descartar la ventana modal.

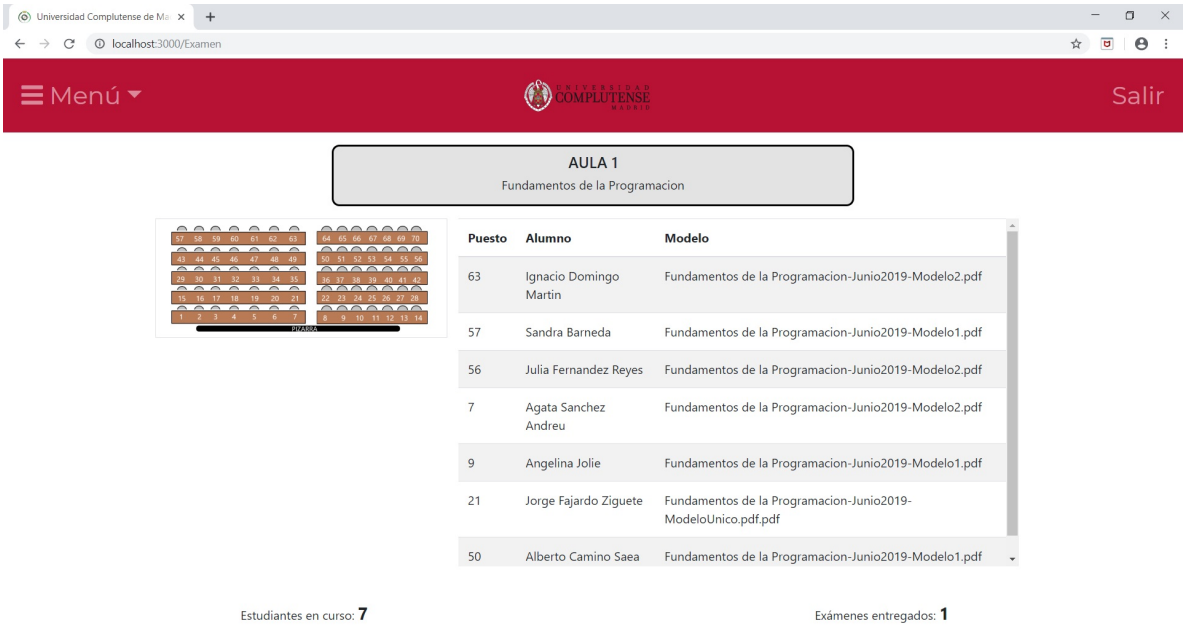

Figura 55: Interfaz de la vista del aula para exámenes en curso. Al clicar en cualquier fila de la tabla de la derecha aparecería el modal con las distintas opciones.

- **exams.ejs:** Esta vista es solo accesible para los profesores. Desde aquí se pueden tanto subir exámenes a la plataforma como visualizarlos desde un visor de PDF o eliminarlos. A la izquierda de la pantalla, se puede observar también una columna con las asignaturas que imparte el profesor que tiene iniciada la sesión en ese momento.

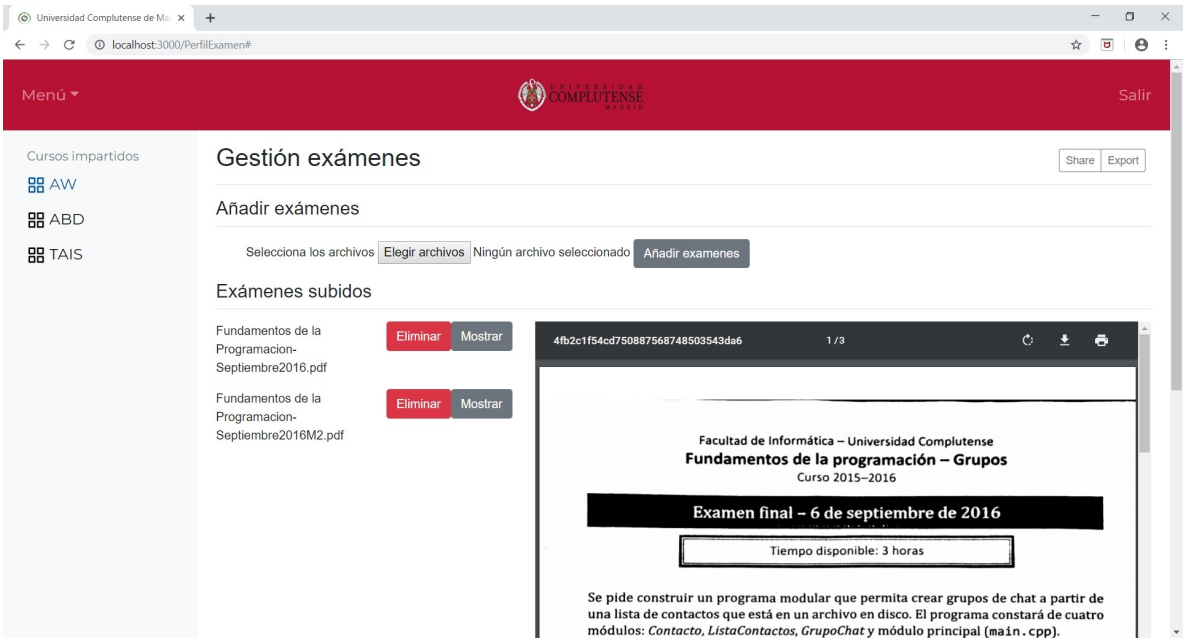

Figura 56: Interfaz de la gestión de exámenes

- **login.ejs:** Es la primera interfaz que se muestra cuando se inicia la aplicación y a la que se redirige siempre que se intenta entrar en otra ruta sin haber antes iniciado sesión. En ella se da la opción de identificarse mediante un usuario y una contraseña, que deben haber sido otorgadas previamente por el administrador del sistema. Solo pueden acceder a la aplicación los profesores, tanto de guardia como los profesores responsables, y los administradores, quedando exentos los alumnos. Dependiendo del tipo de actor universitario, se redirigirá a una vista u otra, en función del rol que desempeña cada uno.

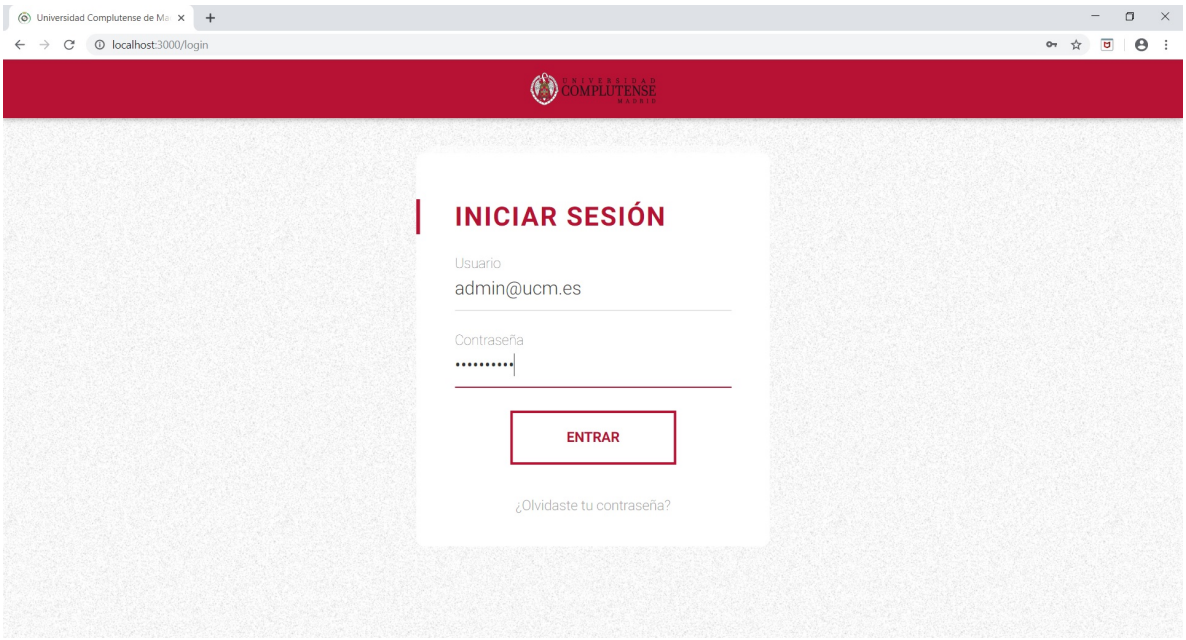

Figura 57: Interfaz de la vista de inicio de sesión.

- **error.ejs**: La interfaz de error recoge todos los posibles casos en los que el servidor devuelva al usuario un código de estado de error 404. Este código de estado indica que alguna redirección de la aplicación o una ruta introducida por el usuario directamente en el navegador se trata de un enlace roto o que ya no existe y que, por lo tanto, no es posible navegar por él.

8.4 Funcionamiento del sistema

8.4.1 Base de datos

Ya que la información del alumnado de la Universidad es confidencial y no se nos ha podido dar acceso a la base de datos de la UCM, para la realización de este proyecto se ha desarrollado una alternativa basándonos en la inclusión de la información que creemos necesaria. El sistema de gestión de base de datos elegido fue MySQL. En este

sistema se guarda toda la información que es necesaria en la aplicación para la subida de exámenes, la administración de fechas de exámenes, administración de profesorado y alumnado y la información necesaria para la generación de las vistas de las aulas con los alumnos.

La estructura de la base de datos se ha realizado de la siguiente manera: en primer lugar, el sistema cuenta con una tabla donde se guardan los actores universitarios que juegan un rol en el funcionamiento de la plataforma; aquellos actores que necesiten credenciales de acceso a la misma, como el profesorado, pueden ser dados de alta, y su información se almacenará en la tabla "conectar". Por otra parte, los profesores se relacionan con las asignaturas que imparten mediante una tabla llamada "imparte", que contiene tanto el id del profesor como el de las asignaturas y el grupo en el que imparte cada asignatura (estando tanto la información de "grupo" como de "asignatura" alojadas en otras dos tablas) Asimismo, estos profesores están también relacionados con la tabla "aviso", que contiene los mensajes de las posibles situaciones de copias que se han dado en los exámenes de su asignatura.

Para organizar los exámenes y sus distintos modelos se tiene una tabla con la información de la capacidad de las aulas, que se relacionan con las tablas "examen" y "modelos" para que el algoritmo de asignación aleatoria trabaje con todas las variables relativas a dichas tablas. También se gestionan las entregas mediante el uso de una tabla donde se guarda la hora a la que entrega cada alumno el examen.

Por último, se tiene una tabla de eventos donde se guardan las fechas de los exámenes junto con el nombre de la asignatura de cada examen. Estos eventos están relacionados con la tabla "examen" mediante una clave ajena en esta última tabla. A continuación se muestra un esquema con las diecisiete tablas que contiene la base de datos y sus relaciones:

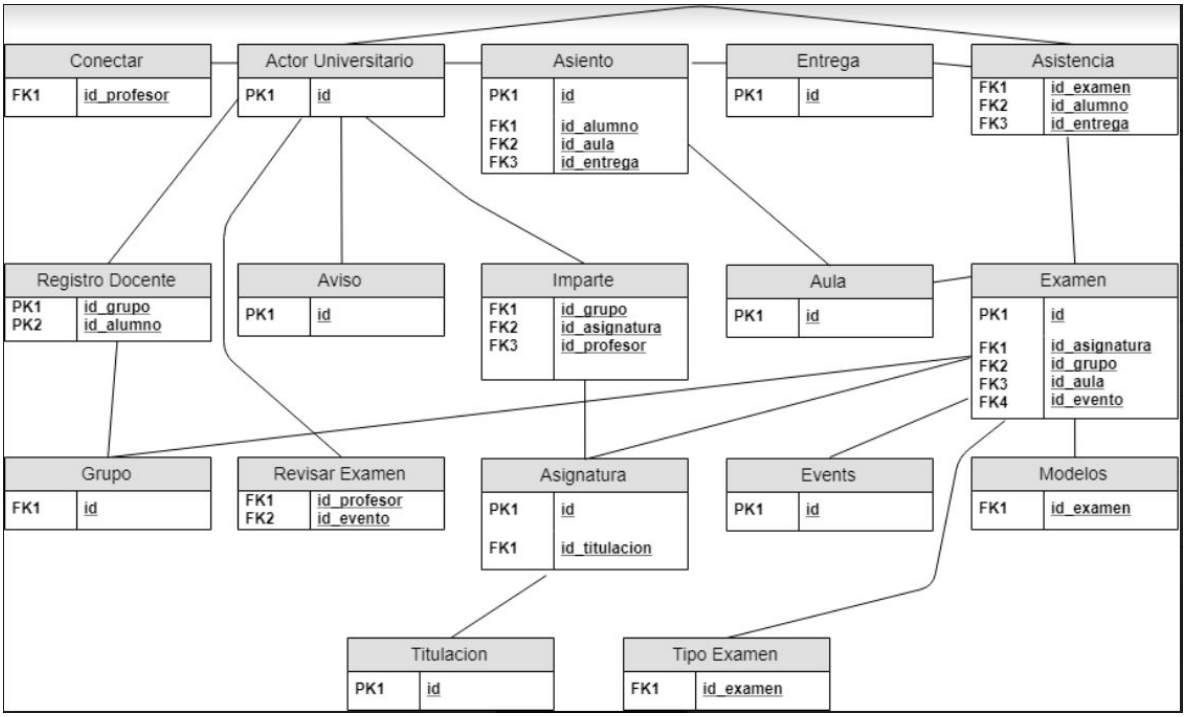

Figura 58: Esquema de la Base de Datos.

Para el correcto funcionamiento de la aplicación, se tiene que tener en cuenta las restricciones mencionadas en el apartado 4.3.1.

8.4.2 Algoritmo de asignación aleatoria

Dentro del DAO que se encarga de garantizar el funcionamiento relativo a la gestión de exámenes podemos encontrar las funciones que permiten que la aplicación sea capaz de garantizar una asignación de puestos aleatorios, que disminuya las posibilidades de copia y automatice la labor del profesor a la hora de repartir los exámenes. Este sencillo algoritmo genera un número de fila y un número de columna aleatorios con valores entre uno y el número total del que disponga el aula en el que se realiza el examen, hasta generar un conjunto de valores que no haya sido asignado con anterioridad. Una vez generados, se genera un array con los distintos modelos de examen de esa asignatura que hayan sido guardados en la base de datos. Este array se rota dependiendo de si la fila es par o impar, con el objetivo de que nunca haya dos exámenes iguales en dos puestos inmediatamente contiguos horizontal o verticalmente. Gracias a esta rotación, se garantiza que tanto filas como columnas cumplan esta condición con independencia de cuantos modelos sean, y sin importar la capacidad del aula.

Una vez se realiza este proceso, la aplicación garantiza que no pueda volver a ser escogido de nuevo aleatoriamente gracias a una cookie de sesión que contiene todos los asientos que ya han sido ocupados.

9 Trabajo individual

En este apartado se va a explicar el trabajo que ha realizado cada uno de los integrantes del equipo, especificando las tareas que se han llevado a cabo, cómo se han realizado y si se produjo algún contratiempo. Para llevar a cabo la organización a lo largo de todo el trabajo, se ha seguido la metodología Scrum, una metodología ágil. Dicha metodología está caracterizada por el seguimiento del trabajo y por realizar cambios en cada iteración, de manera que, si al finalizar una iteración se descubren fallos imprevistos, se pueden corregir y volver a realizar pruebas sobre ellos más adelante. Escogimos Scrum[32] dado que, gracias a que nos proporciona dicha ventaja, nos facilita la mejora de implementación o corrección de errores, ya que se podría cambiar en cualquier momento después de la iteración.

9.1 Ignacio

Tras documentarnos a fondo sobre la temática de nuestro Trabajo de Fin de Grado, mis compañeras y yo decidimos hacer en primer término un reparto inicial de tareas a partir de un índice preliminar que creíamos podía reunir las principales líneas que seguiría el proyecto en el futuro más inmediato. Dentro de este primer reparto, personalmente me tocó encargarme de realizar la introducción y de su consecuente traducción. Para realizarla, tuve que investigar sobre el desarrollo a lo largo del tiempo del método de evaluación de alumnos en los centros de aprendizaje hasta llegar al día de hoy, así como informarme sobre el estado actual de la logística de exámenes tanto en la Universidad como en la Facultad más específicamente. Una vez realizado, me dispuse a explicar la motivación y el objetivo de este proyecto y los problemas que pretendíamos erradicar, así como detallar la estructura de esta memoria para que sirviese de guía para el lector y facilitase su lectura o búsqueda de información.

Una vez depurados los primeros errores, organizamos una segunda reunión grupal para decidir cuáles serían los siguientes pasos a seguir. Con el objetivo de hacer una especificación de requisitos lo más completa posible, decidimos que lo más adecuado sería realizar esa sección todos juntos, discutiendo las ideas y dudas que íbamos planteando tanto yo como mis compañeras Julia y Ágata sobre las necesidades y limitaciones que suponía el potencial entorno que iba a tener nuestra aplicación.

Fue entonces, tras terminar de definir dichos requisitos y al empezar a discutir los posibles riesgos y elaborar los primeros diagramas, cuando detectamos la necesidad de elaborar la memoria en un editor de textos más potente al que estábamos acostumbrados. Una vez Simon nos introdujo en LaTeX[33], encontramos que el editor de textos en línea Overleaf se ajustaba perfectamente a las necesidades de nuestro equipo.

A continuación me dispuse a realizar la elección de tecnologías con las que desarrollaríamos la aplicación; como se explica en esta memoria en el apartado 6.1, la decisión

que se tomó fue la de realizar la aplicación en el entorno de Node.js, haciendo uso de lenguajes como HTML, JavaScript, CSS y bibliotecas como JQuery, que todos conocíamos gracias a la asignatura de Aplicaciones Web.

Tras terminar la configuración inicial de los puertos y los paquetes de Node.js que son necesarios para su correcto funcionamiento, la primera parte de la aplicación que me dispuse a desarrollar fue la vista correspondiente al acceso de usuarios y sus correspondientes funciones del Modelo (DAO usuarios) y los manejadores de ruta en la capa controlador (app.js). Este primer contacto con la aplicación no me requirió demasiado esfuerzo en completar, puesto que habíamos realizado ya otros controles de acceso similares durante la carrera.

Cuando mi compañera Julia tuvo terminada la estructura del diseño de la base de datos que previamente habíamos ideado todos juntos, me dispuse a realizar la parte necesaria para la funcionalidad de administración de exámenes, incluyendo de nuevo tanto la vista como las consecuentes funciones y manejadores del resto de capas. Tras mucha investigación y a base de fallar en numerosas ocasiones a la hora de tratar de conseguir la implementación desde el que considerábamos el enfoque más oportuno para desarrollar esta parte, decidí hacer uso de la API Scheduler de DHTMLX. Este desarrollo me llevó más tiempo del esperado por la aparición de pequeños errores que no supe solucionar hasta que realicé una lectura más profunda de la documentación de dicha herramienta. Al final, gracias a unos pequeños cambios en la BBDD, conseguimos que la funcionalidad acabase ejecutándose tal y como deseábamos.

Proseguí el desarrollo de la aplicación con la implementación de las aulas con la ayuda de Julia, funcionalidad a la que por desgracia no conseguimos darle todo el potencial que queríamos (en un principio, planteamos la posibilidad de desarrollar un mapa dinámico de asientos similar al de aerolíneas o teatros, pero acabamos optando por la inclusión de una imagen estática y una tabla dinámica con la información de los alumnos y las acciones pertinentes al profesorado tales como entregar o notificar copias). Continué con la implementación del algoritmo necesario para la disposición aleatoria de los alumnos en las aulas y la asignación del modelo de examen correspondiente. Este pequeño reto supuso tener que idear un algoritmo que fuera sencillo y que cubriese las necesidades que estábamos buscando, pero finalmente conseguimos el objetivo, no sin haber probado varios enfoques incorrectos. El siguiente paso fue la inclusión de diferentes bibliotecas que incluían mejoras en el diseño de la aplicación, así como la finalización del DAO de administración y los últimos retoques a la hora de implementar, corregir y optimizar diversas partes de las capas controlador y vista.

Una vez teníamos la demo de la aplicación ya operativa y funcionando correctamente, redacté aquellos apartados del Sistema que me había tocado desarrollar. Posteriormente, me dispuse a escribir junto a mis compañeras un último apartado en la memoria en la que incluíamos nuestras conclusiones y las líneas futuras, intentando ofrecer nuestra visión del trabajo realizado y dejando la puerta abierta a diversas funcionalidades que creemos que encajarían perfectamente en la aplicación si se pudieran desarrollar

en un futuro; estas funcionalidades incluyen aquellas que nos hubiera gustar incluir a nosotros desde un principio, pero que por falta de tiempo nos acabó siendo imposible desarrollar.

Con este capítulo terminado, repasamos la memoria completa con el objetivo de encontrar fallos de acuerdo con la normativa y poder corregirlos, así como empezar a trabajar en nuestra presentación y rematar los últimos flecos en cuanto a la maquetación de la misma. Entre los fallos más destacados se encontraba la no traducción de diversas secciones de la memoria, por lo que me encargué de traducirlas al inglés como se exigía en la normativa para los trabajos de fin de grado.

9.2 Julia

Primero organizamos el reparto de tareas inicial. Una vez distribuidas las tareas y puntos a realizar por cada miembro, mi aportación fue hacer el apartado de Antecedentes y plan de trabajo. Para empezar a informarme sobre dicho punto, envié correos electrónicos a diversas universidades de Europa, con el fin de obtener información sobre su método al realizar exámenes. Dadas las insuficientes respuestas obtenidas, tuve que tomar la decisión de investigar más profundamente y cambiar alguna universidad acordada con mis compañeros previamente. Tras la corrección del tutor, añadí el apartado explicando la gestión de exámenes en la UNED, corregí errores y se dio por finalizado este punto.

Una vez terminados la Introducción realizada por Ignacio y el modelo de dominio, organizamos una reunión para proponer y discutir los requisitos que queríamos implantar en la aplicación. Una vez establecidos, posteriormente yo me encargué de definir los grados de complejidad que se pueden desarrollar en la aplicación y los incorporé a la memoria. Más adelante entre Ágata, Ignacio y yo, fuimos corrigiendo errores en los diagramas que había realizado Ágata en un principio con las correcciones y mejoras que nos iba sugiriendo el tutor.

A la par de ir haciendo modificaciones en los diagramas e ir corrigiendo errores, empezamos a pensar los riesgos que podrían aparecer al realizar la aplicación. Tuvimos una reunión donde definimos los riesgos que podrían surgir y realizamos un estudio de éstos. En este apartado, mi aportación fue, una vez hecho el estudio y haber dado las características necesarias a cada uno, realizar las tablas. En la primera tabla (Análisis de riesgos), puse los riesgos con su probabilidad y su efecto. Después en las tablas del apartado Priorización del riesgo, realicé una donde mediante las características de la probabilidad con la que puede ocurrir y el efecto que puede tener, se le otorga un valor a cada uno de ellos. Habiéndoles dado dicho valor, por último, hice la tabla de Análisis y priorización del riesgo, donde se ordenan por gravedad. Después de la revisión del tutor, se tuvo que cambiar un riesgo y realicé el cambio reflejándolo en las tablas.

Cuando este apartado se terminó, y con ellos, gran parte de la memoria, organizamos otra reunión para discutir y elegir las tecnologías que íbamos a usar en la aplicación. Nos

basamos en los conocimientos adquiridos en el grado y la experiencia de cada uno en las diferentes tecnologías que propusimos en un principio. Una vez hecha la reunión, entre los tres hicimos el apartado en la memoria. Una vez terminado dicho apartado y cuando Ágata terminó el apartado de Verificación y validación. Mi aportación fue corregir los errores después de la revisión del tutor y dar los últimos matices al apartado.

Al mismo tiempo de realizar las revisiones y mejoras de los diagramas y los riesgos, empezamos a desarrollar la aplicación. Mi trabajo en este sector fue, en primer lugar, crear las sentencias SQL para la creación de la base de datos, lo que se tuvo que modificar varias veces al cambiarse también los diagramas. También creé sentencias para insertar información en las tablas y poder realizar alguna prueba futura con ellas. Una vez finalizado esto y haciendo correcciones, al mismo tiempo hice junto a Ignacio, crear la vista de las aulas. Hicimos varias vistas de prueba hasta elegir la definitiva. También diseñé junto a Ágata la vista de administrador, la cual no nos dio problemas.

Una vez finalizadas las vistas, la siguiente tarea que hice fue desarrollar junto a mis compañeros el modelo. Mi aportación fue, junto a Ignacio, crear el DAO_teachers. Esta tarea nos llevó mas tiempo del esperado porque era más complicado de lo que creíamos en un principio. El problema principal que observamos fue comprobar si había un examen programado para esa hora cuando un profesor inicia sesión porque si tiene un examen programado a esa hora, se redirige a la vista del aula directamente, pero si no tiene examen, se redirige a la vista de subir un examen a la plataforma.

Por último, cuando terminamos de hacer la implementación se comprobó que todo tenia su funcionalidad bien y se ejecutaba todo correctamente. El inconveniente fue que no nos dio tiempo a realizar las pruebas unitarias ni de sistema mencionadas en el apartado 5, pero cada uno comprobó que la aplicación cumplía los requisitos y funcionaba correctamente.

Una vez finalizado toda la implementación y comprobado que funcionaba correctamente, realicé el apartado de resultados y conocimientos adquiridos con todo aquello que hemos aprendido gracias al haber realizado este proyecto. También junto a Ignacio, se hizo el último apartado con la conclusión y las lineas futuras, donde se muestran más funcionalidades que nos gustaría aplicar al proyecto en un futuro, ya que por falta de tiempo y conocimiento no se ha podido realizar.

Una vez dada por finalizada la memoria, corregimos las partes que necesitaban mejorar o se debía hacer algún cambio y dimos por finalizada tanto la implementación como la documentación.

9.3 Ágata

Como ya han comentado mis compañeros previamente, lo primero que hicimos fue repartirnos los puntos de la memoria, de manera que mi trabajo consistió en crear un

modelo de dominio para la aplicación. En dicho modelo tuve en cuenta las restricciones que había actualmente en el proceso de un examen en la facultad, así como la preparación, la ubicación, los actores que interactuarían en el dominio de la aplicación y el acceso de un alumno a este. Una vez hecho esta investigación e identificando las entidades necesarias para modelar el dominio de la aplicación, empleé la herramienta que facilita IBM-RSA[17] para diagramas UML.

Tras haber realizado varias revisiones y viendo qué podría estar mal o qué mejoras se podrían hacer para llevar a cabo el trabajo de manera exitosa, Ignacio, Julia y yo organizamos una reunión para hablar y proponer los requisitos que tendría nuestra aplicación. En este punto tuvimos en cuenta todas las variables que serían necesarias incluir y/o especificar, teniendo en cuenta la ubicación de los exámenes, el número de exámenes que se podían realizar en un mismo aula y el hardware que sería necesario disponer para poder implementar correctamente la aplicación. Seguidamente, tuve que realizar mi siguiente tarea teniendo en cuenta dichos requisitos.

Esta tercera tarea se comprendía en diseñar los procesos que comprendía la aplicación, en concreto el 'proceso examen' y el 'proceso preparación examen' que ejecutaban los distintos actores. Esto se puede observar con más detalle en el apartado 4.2, donde hay varias imágenes que explican cada proceso. Estos se han realizado mediante el lenguaje estándar BPMN, ya que de esta manera se podía ver más claro cómo funcionaría la aplicación y los pasos que efectuaba cada usuario. Durante varias semanas estuve investigando en varias páginas web propias de las herramientas para poder realizar los procesos. Entre todas las que encontré decidí escoger una denominada Bizagi Modeler[34] porque ofrecía elementos claros, sencillos de comprender, además de que incluía todo lo necesario para poder representar los diagramas que teníamos que modelar. Dicha tarea me llevó más tiempo del esperado dado que no tenía experiencia ni conocimientos sobre la herramienta usada. No obstante, tras varias reuniones con el tutor y varias opciones de diagramas modificados por Julia, por Ignacio y por mi, pudimos dar por finalizados dichos diagramas. Después, teniendo los procesos definidos, realicé un modelo de dominio que recogía todos los artefactos necesarios para la aplicación, así como las distintas entidades que se emplearían en ella y las relaciones y multiplicidades con otras entidades. Esto se explica más detalladamente en el apartado 4.3. Para finalizar este trabajo lo siguiente que tuve que diseñar fue los distintos casos de uso de los actores que intervenían en la aplicación, así como las funciones de cada uno de ellos. Para modelar dichos casos de uso, se emplearon varios diagramas de casos de uso y de secuencia a través del mismo lenguaje aplicado en los diagramas del dominio explicados anteriormente, UML. Todo esto se puede visualizar en el apartado 4.5. Este trabajo requirió mucho esfuerzo, ya que mis conocimientos sobre el lenguaje citado eran básicos y no apliqué en su totalidad las funciones que me facilitaba la herramienta IBM-RSA[17] para hacer los diagramas correspondientes. No obstante, gracias a la dedicación de nuestro tutor y tras muchas revisiones de todos los diagramas dimos con la representación más adecuada y con ello se finalizaron con éxito.

Más adelante, mientras se realizaban las correspondientes revisiones por parte del

tutor, volvimos a planificar otra reunión para tratar el apartado de los riesgos. En este punto tuvimos en cuenta todos los riesgos que podían llegar a surgir durante la realización de todo el trabajo y cómo podíamos solventarlos, mediante un Plan de Riesgos. El trabajo que tuve que realizar en este apartado fue las Técnicas de Reducción, Supervisión y Gestión del Riesgo, el cual empleando la Regla de Pareto, ya mencionada en la sección correspondiente, definí el plan de los cuatro riesgo de mayor gravedad tal y como se puede ver en el apartado 7.7.

Seguidamente, como ya han mencionado mis compañeros, organizamos una reunión para debatir las tecnologías que emplearíamos en nuestra aplicación. Para ello, nos basamos en los conocimientos adquiridos en las respectivas asignaturas de la carrera y la experiencia que teníamos en dichas tecnologías. Una vez estudiado las múltiples opciones escogimos las que más conocíamos y las que más se adaptaran a la aplicación que íbamos a hacer. Después, mi tarea fue realizar el apartado de la memoria de Verificación y Validación. Con lo que ya habíamos debatido se podía llegar a pensar cómo íbamos a realizar esta parte del *testing*, por lo que documenté las pruebas que haríamos en la aplicación para, posteriormente, poder realizarlas. No obstante, al final no se pudieron ejecutar por los motivos mencionados en ese mismo punto.

Llegados a este punto, ya nos pusimos a implementar la aplicación dado que teníamos claro los requisitos, la BBDD que teníamos que usar y las herramientas necesarias para ello. Mi trabajo en la implementación fue hacer la vista de Exámenes y User_admin. Esta última la realizamos en conjunto Julia y yo. La más complicada fue la de Exámenes, ya que tuve que investigar cómo usar un visor de PDF en JavaScript para que se visualizasen los exámenes. No obstante, tras haber investigado en varias páginas web, encontré una solución sencilla que no requería saber demasiado sobre la librería empleada.

Una vez realizada la vista de Exámenes pude ponerme a implementar el DAO_Exams. Esta tarea me llevó más tiempo del esperado debido a que no sabía cómo conectar la librería de PDF con los datos. No obstante, se pudo resolver con éxito. Tras esto, implementé en el controlador, concretamente en el app.js, las partes relacionadas con los exámenes, de modo que se comunicara cada capa correctamente. Posteriormente, seguí aportando conocimientos y trabajo en la parte del controlador, en varios JavaScripts para poder comunicar el trabajo que habían realizado mis compañeros en ambas capas: modelo y vista de las distintas clases, así como la vista del aula y su correspondiente modelo.

Por último, cuando se terminó de hacer la implementación se comprobó que funcionaba todo correctamente. Aunque no nos dio tiempo a realizar las pruebas unitarias ni de sistema como ya mencionamos en el apartado 5, cada uno comprobó que la aplicación hacía lo que debía. Tras esto, corregimos partes de la memoria que no estaban del todo correctas y dimos por finalizada tanto la documentación como la implementación.

10 Resultado y conocimientos adquiridos

Como ya se ha explicado en la introducción, el objetivo de este proyecto es poder optimizar la realización de exámenes, así como mejorar la comunicación entre profesores que los vigilan.

En primer lugar, tras estudiar y sacar información sobre cómo han avanzado a lo largo de la historia los métodos de examinarse, analizamos la gestión de exámenes en la UNED, ya que su método es muy parecido al que queremos desarrollar nosotros en este proyecto. También, buscamos más métodos similares en algunas universidades europeas, dado que los métodos de evaluación son distintos y tienen aspectos en común con el método de la UNED. Todo esto se explica en el apartado 3.

Teniendo ya claro lo que se quería crear, se realizaron los diagramas de cada proceso (apartado 4.2), usando el lenguaje BPMN, para entender mejor el recorrido de la aplicación desde que empieza un examen hasta que finaliza, y las funcionalidades de cada actor universitario durante la realización de éste. Seguido, se realizó el modelo de dominio de la aplicación, donde se puede ver la relación entre las entidades (aparatado 4.3). Con ello se definen los tipos de usuario mediante casos de uso para organizar lo que puede hacer cada usuario dentro de la aplicación. Más tarde se crearon los diagramas de secuencia para cada caso de uso con el fin de mostrar las acciones que emplearía cada usuario que estuviese implicado en la aplicación.

Se estudiaron los riesgos que se podían sufrir y se hizo un análisis de estos.

Una vez hecho todo esto, se comenzó con la implementación. En primer lugar se ha realizado una base de datos normalizada. Se puede observar la estructura en el punto 8.4.1. Dado que no se implementó la opción de pre-inscripción, no se ha podido realizar las funciones propias del modelo estático. No obstante, se ha implementado el modelo dinámico, llegando a alcanzar el grado de complejidad 6, es decir, realizar un examen con distintos modelos en un mismo aula, mediante la simulación de lecturas de tarjetas explicada posteriormente. Para empezar, lo primero que se creó fue la vista de inicio de sesión, donde los usuarios tienen que introducir su usuario y contraseña para poder acceder a la aplicación. Una vez hecho esto, dependiendo del usuario se va a redirigir a una vista o a otra. Las funcionalidades que se han creado para cada usuario son:

- Administrador: Inicia sesión y se redirige a la vista “admin”, donde aparecen todos los actores universitarios y se le permite gestionarlos. A la izquierda de la vista se puede observar una serie de opciones que se pueden realizar. Las que hemos implementado son “Gestionar credenciales”, donde se crean los nombres de usuario con las contraseñas de los usuarios que pueden acceder a la aplicación, “Añadir usuario”, para poder añadir actores universitarios a través del nombre, apellidos y el rol (alumno, profesor..etc.), “Borrar usuario”, para poder eliminar a los actores universitarios y “Gestionar supervisores de exámenes” para organizar los profesores que supervisan cada examen.

- Administrador de exámenes: Cuando inicia sesión con sus credenciales, se le redirige a la vista "useradmin" donde aparece un calendario para añadir los exámenes con su hora y aula. Este usuario crea los eventos de los exámenes, sin ellos los profesores no pueden subir los exámenes de sus asignaturas.

- Profesor responsable: Inicia sesión y se dirige a la vista de "exams" donde tiene la opción de subir exámenes (solo si el administrador de exámenes ha creado el evento de dicho examen). Si ya hay exámenes subidos, tiene la opción de visualizarlos mediante un visor de PDF o eliminarlos si lo desea. En la parte izquierda de esta vista se muestra un menú con las asignaturas que imparte el profesor y una opción para visualizar la vista del aula (solo si tiene un examen en esa hora), ya que el profesor responsable puede vigilar exámenes. Las funcionalidades de la vista del aula se explican en el siguiente usuario (profesor de guardia).

- Profesor de guardia: Una vez inicia sesión, se le redirige directamente a la vista del aula (lab) que tiene que supervisar. En esta vista, a la izquierda se puede ver una imagen del aula con el número de puesto de cada asiento. A la derecha se observa una tabla donde cada fila es la información de un alumno que está realizando el examen en dicho aula. La información que se muestra de cada alumno es el identificador, el nombre y el estado (si sigue realizando el examen o lo ha finalizado). Si seleccionas un alumno, aparece una caja con el nombre y apellidos de dicho alumno y cuatro botones disponibles. Estos son "Entregar", para cuando un alumno finaliza su examen y decide entregarlo, "Notificar posible copia", donde se notifica cualquier incidencia que haya podido ocurrir durante la realización del examen, "Finalizar", esta opción se selecciona cuando un alumno abandona el aula del examen pero ha decidido no entregarlo (siempre que exista esta posibilidad) y "Cerrar", para salir de la visualización de ese alumno en concreto.

Para la posible realización de un examen ha sido necesario crear una simulación de lectura de la tarjeta universitaria mediante una generación aleatoria de alumnos, asignándoles un puesto también aleatorio y un modelo de examen mediante el algoritmo explicado en el apartado 8.4. Por consiguiente, la impresión del examen al pasar la tarjeta se simula enviando el examen de cada alumno a una cola de impresión.

Otro detalle a resaltar es que, para poder desarrollar la aplicación, se han creado varios artefactos que si se diese la opción de llevar a cabo este proyecto en una facultad, no serían necesarios, puesto que se tendría información disponible para realizarlo. Para empezar, la base de datos no sería necesaria crearla al completo, ya que la universidad nos proporcionaría parte de la información en el caso de que el proyecto saliese adelante. También, una aplicación real se conectaría al sistema de 'Inicio de Sesión' de la universidad para el login. Parte del 'admin' estaría ya hecho por la universidad y la otra parte se realizaría usando o extendiendo el esquema de BBDD de la universidad. En definitiva, el resultado de nuestro proyecto ha sido la posibilidad de tener un sistema que abarca todo el proceso de un examen en una facultad, así como poder gestionar los usuarios, los exámenes y las aulas donde se efectúan. El nivel de complejidad que se ha

llegado a cumplir es el segundo (ver los niveles de complejidad en el apartado 4.1), dando la posibilidad de tener en un aula un examen de una asignatura determinada y con varios modelos de ese examen. Además, mediante el algoritmo explicado se ha podido ubicar a cada alumno en el aula de forma que no tenga a su alrededor un compañero con su mismo examen o modelo, si es que lo hubiese. Esto facilita la imposibilidad de copia entre los distintos alumnos.

Por otro lado, realizar este proyecto nos ha dado la oportunidad de poder aplicar conocimientos adquiridos a lo largo del grado en algunas asignaturas específicas como pueden ser:

- Fundamentos de la programación y Tecnología de la programación, que nos han ayudado a adquirir las bases y la forma de pensar de una manera específica y estructurada a la hora de programar.
- Ingeniería del software resultó muy útil a la hora de definir unos requisitos para la aplicación y al realizar diagramas.
- Modelado de software, que nos enseñó a crear un buen modelado de la aplicación al igual que a la creación de los diagramas que hemos necesitado hacer.
- Aplicaciones Web, que nos ha ayudado a crear toda la aplicación en las tecnologías que decidimos usar, ya que nos proporcionó conocimientos sobre HTML, CSS, JavaScript, Bootstrap, JQuery, Node.js y Express.js
- El temario de Base de Datos y Ampliación de Base de Datos nos ha servido para la creación de las sentencias SQL al crear la base de datos y para poder crearlas de una forma normalizada y correcta.
- Gestión de proyectos software, que nos aportó los conocimientos necesarios sobre metodologías que se pueden seguir al realizar un proyecto, la creación de uno siguiendo estas y por tanto la gestión de los riesgos que pueden surgir.

También se han adquirido nuevos conocimientos, entre los que caben destacar:

- BPMN para realizar los diagramas.
- Ampliar el conocimiento sobre UML gracias a la utilización de los estándares de UML 2.0.
- Aprendizaje de LaTeX debido a la necesidad de un editor potente para la realización de esta memoria.
- Uso de librerías externas de JavaScript.
- El uso de la metodología Scrum, ya que un integrante del grupo no tenía experiencia usando esta metodología.

11 Conclusión y líneas futuras

En la actualidad, el método de realización de los exámenes en la facultad de informática no es tan efectivo como creemos que podría ser. La investigación y el desarrollo que hemos llevado acabo tienen como objetivo hacer más eficiente este proceso, optimizando las pérdidas de tiempo y permitiendo tener una mejor organización dentro de las aulas. Ocasionalmente, optimizando la pérdida de tiempo y tener una mejor asignación de aulas. Además, queríamos desarrollar una tecnología que ahorrara una gran cantidad de espacio, tiempo y posibles situación de copia entre los alumnos examinados. Todo el software creado es libre y está bajo licencia GNU GPL.

La aplicación consta de un algoritmo que se encarga de que dos alumnos con el mismo examen en el mismo aula no puedan sentarse uno al lado del otro. Este problema, en la actualidad, al hacerse cargo el profesor de manera manual, crea una pérdida de tiempo a la hora de comenzar un examen. Asimismo, si un alumno llega tarde, el profesor de guardia tiene que mirar los exámenes que están realizando los alumnos de alrededor del sitio en que se va a sentar el nuevo alumno y darle el modelo correspondiente, problema que se va a evitará con la nueva tecnología que queremos implantar.

La aplicación web también consta de un apartado de incidencias en la interfaz de los profesores de guardia, para que si en el caso de que ocurra algún problema con un alumno o se quiera resolver una duda con el profesor responsable, el proceso sea más rápido mediante el uso de unas notificaciones y se evite la pérdida de tiempo tanto para el profesor como para los alumnos. Esto es una gran ventaja, dado que si ocurre algún problema grave, se puede resolver de manera más efectiva.

En resumen, este proyecto nos ha dado la oportunidad de fijarnos unos objetivos y afrontar el reto de poder cumplir dichos objetivos, afrontando dificultades y valorando la importancia de una buena planificación en la gestión de proyectos software.

Con el fin de mejorar el proyecto en algunos aspectos que pensamos que son importantes para una mayor efectividad, consideramos que, a grandes trazos, como líneas de desarrollo para el futuro se podrían implementar los siguientes puntos, los cuales nos ha sido imposible desarrollar por falta de tiempo:

- Añadir un lector de tarjetas para poder pasar la información del alumno mediante este elemento.
- Implantar impresoras en las aulas para que se pudiese imprimir el examen de cada alumno en el momento en el que se presenta a dicho examen imprimiendo su puesto y datos en el enunciado.
- Crear una interfaz de mantenimiento donde, por ejemplo, si uno o varios ordenadores del laboratorio donde se va a realizar un examen está estropeado o no actualizado, no se pueda seleccionar para que un alumno se siente.

- Implantar un lector con pantalla o, en su defecto, un monitor o pantalla de ordenador, para que en el caso de que un alumno quiera acudir a un aula donde ya no haya asientos disponibles o se equivoque de aula, informe sobre el error y muestre una solución. Por ejemplo, si un alumno tiene un examen donde van a usarse dos aulas y quiere acceder a un aula que está llena, en la pantalla se muestre que tiene que ir al otro aula disponible.
- Paralelamente al punto anterior también se podría conseguir que al realizar exámenes en el laboratorio se pueda evitar la impresión en papel. En la pantalla del lector saldría el puesto asignado del alumno y podrían descargarse el enunciado una vez sentados.
- Activar inhibidores en las aulas para evitar copias y acceso a internet.
- Añadir escáneres en las aulas para que los profesores puedan crear una copia de los exámenes de los alumnos y así darles la oportunidad de poder corregir fuera de la Facultad y evitar casos de pérdida. Otra ventaja que se tiene, es la posibilidad de que el profesor pueda enviar a los alumnos una copia electrónica del examen para tener más facilidades a la hora de ir a la corrección. Y por último, se podría generar una copia de seguridad de los exámenes de todos los años por si acaso en un futuro se necesitan.
- Permitir que los alumnos de un examen puedan empezarlo en horas ligeramente distintas y notificar al profesor por medio de la interfaz cuándo acaba cada alumno.
- Implementar la opción de que el profesor responsable de cada asignatura pueda elegir el número de puestos vacíos entre los alumnos (si lo deseara) para la realización del examen.

12 Conclusion and future development

At the present time, the method of conducting exams in the Computer Science Faculty is not as efficient as we consider it could be. The research and development we have carried out have the aim to improve the effectiveness of this process, optimizing the loss of time and have a better classrooms allocation. Moreover, we wanted to develop a technology that would save enourmous amount of space, time and possible cheating situation among the examined students. All software developed is free software and is under GNU GPL License.

The application contains an algorithm that ensures that two students with the same test model within the same classroom can not sit next to each other. This issue is currently solved by the teacher manually, wasting time when starting a test. Additionally, if a student is late, the teacher on duty has to take a look to the exams that are already being taken by the students around the place the new student is about to sit in, and

give him the corresponding model, a problem that will be avoided with the system that we implemented.

The web application also includes a section of incidents in the interface for the teachers on duty, so that if there is any problem with a student or if they need the main teacher for any matter, the process will be faster through some notifications and avoid the loss of time for both students and head teacher. This implies a great advantage, since if a serious issue takes place, it can be solved more effectively.

Altogether, this project has given us the opportunity to set certain goals and face the challenge of being able to meet their deadlines, facing difficulties and letting us appraise the importance of planning within software projects.

In order to improve the project in some aspects that we think are relevant for greater effectiveness, we consider that, in broad brushstrokes, some of the future development could consist in the following points, which we have been unable to develop due to lack of time:

- Add card readers so the students information can be parsed through them.
- Create an additional interface where, for example, if one or more computers in the laboratory where an exam is to be carried out is broken or not updated, it can not be selected by the algorithm as an allocation place.
- Install card readers with screen or, alternatively, any other device with a display, so that in the event that a student wants to go to a classroom where there are no seats available anymore or the classroom he just entered is wrong, the screen would report the error and provide with a solution. For example, if a student is about to take an exam where two classrooms are going to be used, and he or she wants to access a classroom that is already full, the screen would show the classroom in which there is still room for more students.
- In tandem with this last point, it could also be possible to avoid printing on paper when carrying out exams in the laboratories. The randomly assigned position of the student would appear on the reader's screen, and the statement could then be downloaded once the student is seated.
- Activate signal inhibitors in the classrooms when exams start to avoid copies and irregular access to the internet.
- Introduce scanners in the classrooms so that teachers can print backup copies of the exams, given them the chance to correct those exams out of the Faculty and preventing possible cases of loss.
- Allow students within the same exam to start taking it at slightly different times and notify the teacher through the interface when each student's time finishes.

- Deploy the functionality which would allow the responsible teacher for each subject to choose, if desired, the amount of empty spaces between the students when taking the exam.

Siglas

BBDD base de datos. 22, 23, 32, 38, 52–54, 57, 60, 63, 64, 67, 68, 71, 73, 75–78

BPMN Business Process Model and Notation. 24, 74, 76, 78

CRUD Create, Read, Update and Delete. 54, 63

DNI Documento Nacional de Identidad. 15, 22, 24

GPL General Public License. 79

IBM-RSA IBM Rational Software Architecture Designer. 36, 74

IT Tecnología de la Información. 8, 11

MVC Modelo Vista Controlador. 61

RSGR Reducción, Supervisión y Gestión del Riesgo. 56, 60

SQAS-SEI Software Quality Assurance Subcommittee. 58, 59

TFG Trabajo de Fin de Grado. 2, 9, 10, 70

UCM Universidad Complutense de Madrid. 8, 14, 16, 67

UML Unified Modeling Language. 38, 74, 78

UNED Universidad Nacional de Educación a Distancia. 13, 14, 55, 72, 76

Glosario

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. 54, 78

CSS es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado, en este caso HTML. 54, 71, 78

Express.js es un marco de aplicación web para Node.js, lanzado como software gratuito y de código abierto bajo la Licencia MIT. Está diseñado para construir aplicaciones web y APIs. 54, 64, 78

HTML (HyperText Markup Language en inglés) hace referencia al lenguaje de marcado para la elaboración de páginas web. 54, 64, 71, 78

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. 54, 64, 71, 75, 78

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. 54, 71, 78

LaTeX es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas. 70, 78

mySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual. 54, 67

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono y basado en el motor V8 de Javascript. 2, 10, 13, 52, 54, 62, 64, 71, 78

SQL es un lenguaje específico del dominio utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales. 54, 73, 78

Referencias

[1] Alex Usher. *A Brief History of Exams*. Oct. de 2016. URL: http://higheredstrategy.com/a-brief-history-of-exams/.

[2] Professor Derk Bodde. *Chinese ideas in the west*. 2004. URL: http://afe.easia.columbia.edu/chinawh/web/s10/ideas.pdf.

[3] La Valija Virtual. *UNED*. URL: http://portal.uned.es/portal/page?_pageid=93,36662434&_dad=portal&_schema=PORTAL (visitado 05-2019).

[4] La Valija Virtual. *UNED Examenes*. URL: http://portal.uned.es/portal/page?_pageid=93,53408118&_dad=portal&_schema=PORTAL (visitado 05-2019).

[5] La Valija Virtual. *UNED*. URL: http://www.dicub.es/p18/valija-virtual.aspx (visitado 05-2019).

[6] Tallin University. *Examinations and Assessments*. URL: https://www.tlu.ee/en/taxonomy/term/90/examinations-and-assessments (visitado 05-2019).

[7] University of Bergen. *Exam registration on Studentweb*. URL: https://www.uib.no/en/student/49276/exam-registration-studentweb (visitado 05-2019).

[8] Univerity Of Helsinki. *Examinations*. URL: https://www.helsinki.fi/en/open-university/studying/during-your-studies/examinations (visitado 05-2019).

[9] Amanda Malmquist. *Exams at Malmö University*. URL: https://www.mah.se/english/Student/For-your-studies/Examination/ (visitado 05-2019).

[10] Martin Jensen. Personal Communication. University of Denmark. 2 de nov. de 2018.

[11] Univerity Of Oulu. *General Exam*. URL: http://www.oulu.fi/university/node/34979 (visitado 05-2019).

[12] Marcial Córdoba Padilla. *Formulación y evaluación de proyectos*. Ecoe ediciones, 2011.

[13] Bangalore D Nagesh Kumar IISc. *Optimization Methods: Linear Programming-Simplex Method-I*. URL: https://nptel.ac.in/courses/105108127/pdf/Module_3/M3L3_LN.pdf (visitado 05-2019).

[14] Bussiness Process Model y Notation. *BPMN*. URL: https://www.omg.org/spec/BPMN/2.0/PDF (visitado 05-2019).

[15] IBM Community. *Modelos de dominio*. URL: https://www.ibm.com/support/knowledgecenter/es/SS9UM9_9.1.2/com.ibm.datatools.logical.ui.doc/topics/cdommod.html (visitado 05-2019).

[16] IBM Community. *Casos de uso*. URL: https://www.ibm.com/support/knowledgecenter/es/SSWMEQ_4.0.6/com.ibm.rational.rrm.help.doc/topics/c_uc.html (visitado 05-2019).

[17] IBM Community. *IBM Rational Software Architect Designer*. URL: https://www.ibm.com/us-en/marketplace/rational-software-architect-designer (visitado 05-2019).

[18] IBM Community. *Diagramas de secuencia y caso de uso de Rational DOORS*. URL: https://www.ibm.com/support/knowledgecenter/es/SSQQC5_6.0.0/com.ibm.rational.sse.doc/topics/rhp_c_int_doors_uc_seq_diags.html (visitado 05-2019).

[19] Unified Modeling Language 2.0. *UML*. URL: https://www.omg.org/spec/UML/2.5.1/PDF (visitado 05-2019).

[20] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. Addison-Wesley, 1992.

[21] Wayne W Eckerson. "Three tier client/server architecture: Achieving scalability, performance and efficiency in client server applications". En: *Open Information Systems* 10.1 (1995).

[22] Maximiliano Cristiá. "Introducción al testing de software". En: *Recuperado el* 14 (2009).

[23] Per Runeson. "A survey of unit testing practices". En: *IEEE software* 23.4 (2006), págs. 22-29.

[24] Mocha Community. *Mocha*. URL: https://mochajs.org/ (visitado 05-2019).

[25] Martyn A Ould y Charles Unwin. *Testing in software development*. Cambridge University Press, 1986.

[26] Manuel Cillero. *Pruebas del Sistema*. URL: https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/sistema/ (visitado 05-2019).

[27] Roy Miller y Christopher T Collins. "Acceptance testing". En: *Proc. XPUniverse* 238 (2001).

[28] Wikiversidad. *Gestión de riesgos de proyectos software*. URL: https://es.wikiversity.org/wiki/Gesti%C3%B3n_de_riesgos_de_proyectos_software#Estrategia_reactiva (visitado 05-2019).

[29] Scott Thompson. "Difference Between a Proactive & a Reactive Business Strategy". En: *Houston Chronicle* (2015).

[30] Judith A Clapp y col. *Software Quality Control, Error, Analysis*. William Andrew, 1995.

[31] Ankunda R Kiremire. "The application of the pareto principle in software engineering". En: *Consulted January* 13 (2011), pág. 2016. URL: http://www2.latech.edu/~box/ase/papers2011/Ankunda_termpaper.PDF.

[32] Ken Schwaber. "Scrum development process". En: *Business object design and implementation*. Springer, 1997, págs. 117-134.

[33] Leslie Lamport. *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley, 1994.

[34] *Bizagi Modeler: Software de modelamiento de procesos de negocios (BPM)*. URL: https://www.bizagi.com/es/productos/bpm-suite/modeler.